Edge Insights for Autonomous Mobile Robots (EI for AMR) Developer Guide

Contents

Chapter 1: Edge Insights for Autonomous Mobile Robots
How it Works 4
Introduction to edgesoftware Command Line Interface (CLI) 10
Edge Insights for Autonomous Mobile Robots Tutorials 17
Robot Tutorials 17
Run a ROS 2 Sample Application in the Docker* Container 17
Run an Intel® RealSense™ ROS 2 Sample Application in Docker* Container 20
Run a GStreamer* Video Pipeline using GStreamer* Plugins in Docker* Container 22
Run a GStreamer* Audio Pipeline using GStreamer* Plugins in Docker* Container 24
Run a GStreamer* Video Pipeline using libv4l2 in Docker* Container 25
Run a GStreamer* Video Pipeline using the Intel® RealSense™ Plugin in the Docker* Container 26
Optional (Only for Sony* IMX390 Setups): Run a GStreamer* Video Pipeline using Sony's IMX390 MIPI Sensor in Docker* Container 27
Run OpenVINO™ Sample Applications in Docker* Container 30
Run ROS 2 OpenVINO™ Toolkit Sample Applications in Docker* Container 34
Run Intel® oneAPI Base Toolkit Sample Applications in Docker* Container 37
Run Profiling Application in Docker* Container with VTune™ Profiler 40
Run OpenVINO™ Benchmarking Tool 43
Run the ADBSCAN Algorithm 46
Launch Wandering Application with Gazebo* Simulation 49
Launch Wandering Application on AAEON* Robot Kit 51
Launch Cartographer with 2D LIDAR 63
Run FastMapping Algorithm 71
Run ROS 2 Navigation Sample Applications Using the ITS Path Planner Plugin in a Docker* Container 73
ITS Path Planner Plugin Customization 77
Run the Edge Insights for Autonomous Mobile Robots Container in KVM Guest 78
Run a Collaborative SLAM System 83
Build New and Custom Docker* Images from the Edge Insights for Autonomous Mobile Robots SDK 84
Troubleshooting for Robot Tutorials 88
Fleet Tutorials 91
Basic Fleet Management 91
Device Onboarding End-to-End Use Case 107
OTA Updates 123
Wandering Application Deployment 138
Troubleshooting for Robot Orchestration Tutorials 141
Intel® Edge Software Device Qualification (Intel® ESDQ) for EI for AMR 151

Security 157
Real-Time Support 161
Terminology 162
Notices and Disclaimers 163

Edge Insights for Autonomous Mobile Robots

1

Edge Insights for Autonomous Mobile Robots (EI for AMR) offers containerized software packages and pre-validated hardware modules for sensor data ingestion, classification, environment modelling, action planning, action control. Based on the Robot Operating System 2 (ROS* 2), it also includes the OpenVINO™ toolkit, Intel® oneAPI Base Toolkit (Base Kit), Intel® RealSense™ SDK, and other software dependencies in a container, along with the source code, as well as reference algorithms and deep learning models as working examples.

In addition to autonomous mobility, this package showcases map building and Simultaneous Localization And Mapping (SLAM) loop closure functionality. The package uses an open source version of visual SLAM with camera input from an Intel® RealSense™ camera. Optionally, the package allows you to run Light Detection and Ranging (LIDAR) based SLAM and compare those results with visual SLAM results on accuracy and performance indicators. In addition, this package detects the objects and highlights them in the map. Depending on the platform that is used, AI workloads are run on an integrated GPU or on Intel® Movidius™ Myriad™ X accelerator.

Edge Insights for Autonomous Mobile Robots helps to address various industrial and manufacturing uses, consumer market and smart cities use cases, which include data collection, storage, and analytics on a variety of hardware nodes across the factory floor. See How it Works.

Use the Get Started Guide for Robots and Get Started Guide for Robot Orchestration for installation instructions.

For an introduction to the edgesoftware command line interface for managing Intel's Developer Catalog packages, see Introduction to edgesoftware Command Line Interface (CLI).

See Edge Insights for Autonomous Mobile Robots Tutorials for step-by-step, hands-on walkthroughs, including how to run a demo ROS 2 sample application inside the EI for AMR Docker* container and set up basic fleet management.

How it Works

The Edge Insights for Autonomous Mobile Robots (EI for AMR) modules are deployed via Docker* containers for enhanced Developer Experience (DX), support of Continuous Integration and Continuous Deployment (CI/CD) practices and flexible deployment in different execution environments, including robot, development PC, server, and cloud.

This section provides an overview of the modules and services featured with Edge Insights for Autonomous Mobile Robots.

Modules and Services

The middleware layered architecture in the Intel® oneAPI Base Toolkit (Base Kit) and Intel® Distribution of OpenVINO™ toolkit (OpenVINO™) abstracts hardware dependencies from the algorithm implementation.

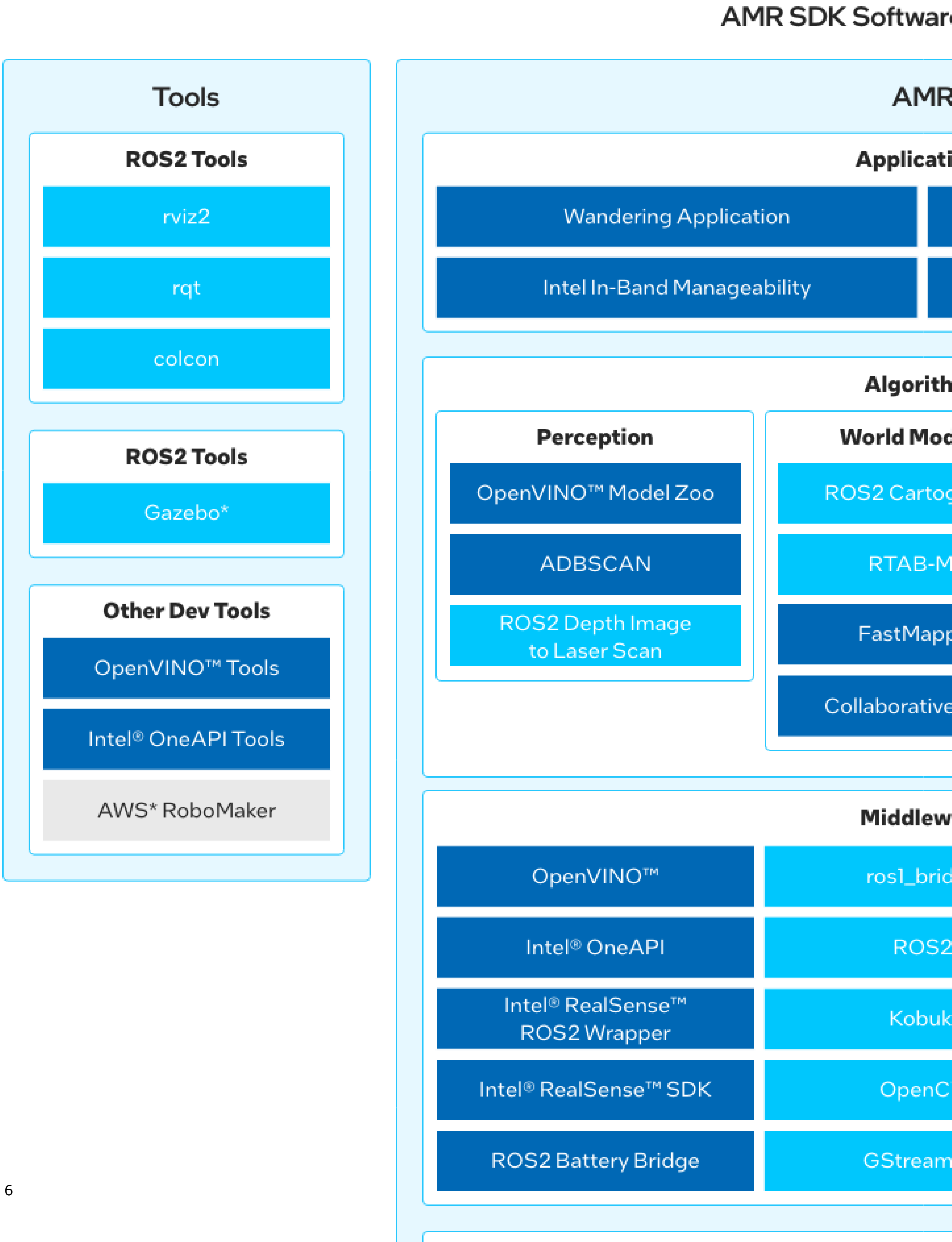
The ROS 2 with data distribution service (DDS) is used as a message bus. This Publisher-Subscriber architecture based on ROS 2 topics decouples data providers from consumers.

Camera and LIDAR sensor data is abstracted with ROS 2 topics.

Video streaming processing pipelines are supported by GStreamer*. It decouples sensor ingestion, video processing and AI object detection via OpenVINO™ toolkit DL Streamer framework.

Also, more complex computational graphs that decouple Sense-Plan-Act autonomous mobile robot applications can be implemented using ROS 2 topic registration.

This diagram shows the software components included in the EI for AMR package. The software stack keeps evolving iteratively with additional algorithms, applications, and third-party ecosystem software components.


AMR SDK Softwar
Tools
ROS2 Tools
rviz2
rqt
colcon
ROS2 Tools
Gazebo*
Other Dev Tools
OpenVINO™ Tools
Intel® OneAPI Tools
AWS* RoboMaker
AMR
Applicati
Wandering Application
Intel In-Band Manageability
Algorith
Perception
OpenVINO™ Model Zoo
ADBSCAN
ROS2 Depth Image to Laser Scan
World Mod
ROS2 Cartog
RTAB-M
FastMapp
Collaborative
Middlew
OpenVINO™
Intel® OneAPI
Intel® RealSense™ ROS2 Wrapper
Intel® RealSense™ SDK
ROS2 Battery Bridge
ros1_brid
ROS2
Kobuk
OpenC
GStream

The EI for AMR software stack is based on software supported by and part of the underlying hardware platform, their respective Unified Extensible Firmware Interface (UEFI) based boot, and their supported Linux* operating system. For requirement details, see:

- Get Started Guide for Robots
- Get Started Guide for Robot Orchestration

EI for AMR Drivers

Edge Insights for Autonomous Mobile Robots relies on standard Intel® Architecture Linux* drivers included and upstreamed in the Linux* kernel from kernel.org and included in Ubuntu* distributions. These drivers are not included in the EI for AMR package. Some notable drivers that are specifically important for EI for AMR include:

- 5G/LTE Device Drivers for 5G and LTE connectivity.
- Battery Bridge Kernel Module, which allows user-space applications to feed battery and power information into the Linux kernel's power supply subsystem. It has been designed to be used together with the ROS 2 Battery Bridge to allow ROS 2-based AMR software stacks to forward battery information from an AMR's microcontroller into the Linux kernel.
- MIPI CSI IMX390 Device Driver, for cameras that are using the Sony* IMX390 sensor and are connected to a Tiger Lake platform SoC via a MIPI CSI connection.
- Device Drivers for Intel® Movidius™ Myriad™ X VPUs.

EI for AMR Middleware

EI for AMR integrates the following middleware packages on AMRs:

- AAEON* ROS 2 interface, the ROS 2 driver node for AAEON AMRs
- GStreamer*, which includes support for libv4l2 video sources, GStreamer* "good" plugins for video and audio, and GStreamer* plugin for display to display a video stream in a window
- Kobuki, the ROS 2 driver node for Cogniteam's Pengo AMRs
- Intel® oneAPI Base Toolkit, which is able to execute Intel® oneAPI Base Toolkit sample applications. The Intel® oneAPI Base Toolkit is a core set of tools and libraries for developing high-performance, data-centric applications across diverse architectures. It features an industry-leading C++ compiler and the Data Parallel C++ (DPC++) language, an evolution of C++ for heterogeneous computing. For Intel® oneAPI Base Toolkit training and a presentation of the CUDA* converter, refer to:
  - Intel® DPC++ Compatibility Tool Self-Guided Jupyter Notebook Tutorial
  - Optimize Edge Compute Performance by Migrating CUDA* to DPC++
- OpenCV (Open Source Computer Vision Library), an open-source library that includes several hundred computer vision algorithms
- Intel® Distribution of OpenVINO™ toolkit, which is a comprehensive toolkit for quickly developing applications and solutions that solve a variety of tasks including emulation of human vision, automatic speech recognition, natural language processing, recommendation systems, and many others. Based on latest generations of artificial neural networks, including Convolutional Neural Networks (CNNs), recurrent and attention-based networks, the toolkit extends computer vision and non-vision workloads across Intel hardware, maximizing performance. It accelerates applications with high-performance, AI and deep learning inference deployed from edge to cloud.
- Intel® RealSense™ ROS 2 Wrapper node, used for Intel® RealSense™ cameras with ROS 2
- Intel® RealSense™ SDK, used to implement software for Intel® RealSense™ cameras
- ROS 2 ros1_bridge, which provides a network bridge allowing the exchange of messages between ROS1 and ROS 2. This lets users evaluate the EI for AMR SDK on AMRs or with sensors for which only ROS1 driver nodes exist.
- ROS 2, Robot Operating System (ROS), which is a set of open source software libraries and tools for building robot applications

ROS 2 depends on other middleware, like the Object Management Group (OMG) Data Distribution Service (DDS) connectivity framework that is using a publish-subscribe pattern. The standard ROS 2 distribution includes eProsima Fast DDS implementation.

- ROS 2 Battery Bridge, which utilizes the Battery Bridge Kernel Module to forward battery information from an AMR's microcontroller into the Linux kernel
- RPLIDAR ROS 2 Wrapper node, for using RPLIDAR LIDAR sensors with ROS 2
- SICK Safetyscanners ROS 2 Driver, which reads the raw data from the SICK Safety Scanners and publishes the data as a laser_scan msg

EI for AMR Algorithms

Edge Insights for Autonomous Mobile Robots includes reference algorithms as well as deep learning models as working examples for the following automated robot control functional areas:

- DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is an unsupervised clustering algorithm that clusters high dimensional points based on their distribution density. Adaptive DBSCAN (ADBSCAN) has clustering parameters that are adaptive based on range and are especially suitable for processing LIDAR data. It improves the object detection range by 20-30% on average.
- Collaborative SLAM, a collaborative visual simultaneous localization and mapping (SLAM) framework for service robots. With an edge server maintaining a map database and performing global optimization, each robot can register to an existing map, update the map, or build new maps, all with a unified interface and low computation and memory cost. The Collaborative SLAM system consists of at least two elements:
    - The *tracker* is a visual SLAM system with support for inertial and odometry input. It estimates the camera pose in real-time, and maintains a local map. It can work without a server, but if it has one configured, it will communicate with the server to query and update the map. The tracker represents a robot. There can be multiple trackers running at the same time.
    - The *server* maintains the maps and communicates with all trackers. For each new keyframe from a tracker, it detects possible loops, both intra-map and inter-map. Once detected, the server will perform map optimization or map merging, and distribute the updated map to corresponding trackers.

  For collaborative SLAM details, refer to A Collaborative Visual SLAM Framework for Service Robots paper.
- FastMapping, which is an algorithm to create a 3D voxel map of a robot's surrounding, based on Intel® RealSense™ depth sensor data.
- OpenVINO™ Model Zoo, optimized deep learning models and a set of demos to expedite development of high-performance deep learning inference applications. A developer can use these pre-trained models instead of training their own models to speed-up the development and production deployment process.
- ROS 2 Cartographer, a system that provides real-time simultaneous localization and mapping (SLAM) based on real-time 2D LIDAR sensor data. It is used to generate as-built floor plans in the form of occupancy grids.
- ROS 2 Depth Image to Laser Scan, which converts a depth image to a laser scan for use with navigation and localization.
- ROS 2 Navigation stack, which seeks a safe way to have a mobile robot move from point A to point B. This will complete dynamic path planning, compute velocities for motors, detect and avoid obstacles, and structure recovery behaviors. Nav2 uses behavior trees to call modular servers to complete an action. An action can be to compute a path, control effort, recovery, or any other navigation related action. These are separate nodes that communicate with the behavior tree (BT) over a ROS action server.
- RTAB-Map (Real-Time Appearance-Based Mapping), a RGB-D, Stereo and Lidar Graph-Based SLAM approach based on an incremental appearance-based loop closure detector. The loop closure detector uses a bag-of-words approach to determinate how likely a new image comes from a previous location or a new location. When a loop closure hypothesis is accepted, a new constraint is added to the map's graph, then a graph optimizer minimizes the errors in the map. A memory management approach is used to limit the number of locations used for loop closure detection and graph optimization, so that real-time constraints on large-scale environnements are always respected. RTAB-Map can be used alone with a handheld Kinect, a stereo camera or a 3D lidar for 6DoF mapping, or on a robot equipped with a laser rangefinder for 3DoF mapping.

EI for AMR Applications

- Intel In-Band Manageability, monitors device(s) and updates software and firmware of the device(s) remotely.
- Object Detection AI Application, detects objects in video data using a deep learning neural network model from the OpenVINO™ Model Zoo.
- VDA5050 Sample Handler, processes selected commands from the VDA5050 AMR/AGV interoperability standard and forwards the AMR's software components for autonomous navigation.
- Wandering Application, included in the EI for AMR SDK to demonstrate the combination of the middleware, algorithms, and the ROS 2 navigation stack to move a robot around a room avoiding hitting obstacles, updating a local map in real time exposed as ROS topic, and publish AI-based objects detected in another ROS topic. It uses the robot's sensors and actuators that are available from the robot's hardware configuration.

Edge Server Middleware

- FIDO Device Onboarding, an automatic onboarding protocol for IoT devices. Permits late binding of device credentials, so that one manufactured device may onboard, without modification, to many different IoT platforms.
- Intel® Smart Edge Open, a software toolkit for building edge platforms. It speeds up development of edge solutions that host network functions alongside AI, media processing, and security workloads with reference solutions optimized for common use cases powered by a Certified Kubernetes* cloud native stack.

Edge Server Algorithms

- OpenVINO™ Model Zoo, which includes optimized deep learning models and a set of demos to expedite development of high-performance deep learning inference applications. A developer can use these pre-trained models instead of training their own models to speed-up the development and production deployment process.

Edge Server Applications

- OpenVINO™ Model Server (OVMS), a high-performance system for serving machine learning models. It is based on C++ for high scalability and optimized for Intel solutions, so that you can take advantage of all the power of the Intel® Xeon® processor or Intel's AI accelerators and expose it over a network interface. OVMS uses the same architecture and API as TensorFlow Serving, while applying OpenVINO for inference execution. Inference service is provided via gRPC or REST API, making it easy to deploy new algorithms and AI experiments.
- ThingsBoard*, an open-source IoT platform for data collection, processing, visualization, and device management. It enables device connectivity via industry standard IoT protocols - MQTT, CoAP and HTTP and supports both cloud and on-premises deployments.

Tools

ROS Tools

Edge Insights for Autonomous Mobile Robots is validated using ROS 2 nodes. ROS 1 is not compatible with EI for AMR components. A ROS 1 bridge is included to allow EI for AMR components to interface with ROS 1 components.

- From the hardware perspective of the supported platforms, there are no known limitations for ROS 1 components.
- For information on porting ROS 1 applications to ROS 2, here is a guide from the ROS community.

Edge Insights for Autonomous Mobile Robots includes:

- colcon (collective construction), a command line tool to improve the workflow of building, testing, and using multiple software packages. It automates the process, handles the ordering, and sets up the environment to use the packages.
- rqt, a software framework of ROS 2 that implements the various GUI tools in the form of plugins.
- rviz2, a tool used to visualize ROS 2 topics.

Simulation

Edge Insights for Autonomous Mobile Robots includes:

- The Gazebo* robot simulator, making it possible to rapidly test algorithms, design robots, perform regression testing, and train AI systems using realistic scenarios. Gazebo offers the ability to simulate populations of robots accurately and efficiently in complex indoor and outdoor environments.
- An industrial simulation room model for Gazebo*, the Open Source Robotics Foundation (OSRF) Gazebo Environment for Agile Robotics (GEAR) workcell that was used for the ARIAC competition in 2018.

Other Tools

Edge Insights for Autonomous Mobile Robots includes:

- Intel® oneAPI Base Toolkit, which includes the DPC++ compiler and compatibility tool, as well as debugging and profiling tools like VTune™ Profiler, etc. (formerly known as Intel System Studio).
- OpenVINO™ Tools, including the model optimization tool.

Deployment

All application, algorithm, and middleware components which are executed as standalone processes are deployed in their own Docker container. This allows to selectively pull these components onto an AMR or Edge Server and launch them there.

For development purposes, the middleware libraries and all tools are deployed in a single container called `Full SDK`. This container is constructed hierarchically by extending the `OpenVINO SDK` container, which itself extends the `ROS2 SDK` container. For storage space savings, you can choose to run any of the containers depending on the needs of your application.

- The `ROS2 SDK` container includes the ROS 2 middleware and tools, Intel® RealSense™ SDK and ROS 2 wrapper, GStreamer* and build tools, ROS 2 packages (Cartographer, Navigation, RTAB_MAP) and the Fast Mapping application (the Intel-optimized version of octomap).
- The `OpenVINO SDK` container includes the `ROS2 SDK`, as well as the OpenVINO™ development toolkit, the OpenVINO™ DL GStreamer* plugins and the Wandering demonstration application.
- The `Full SDK` container includes the OpenVINO™ container, as well as the Intel® oneAPI Base Toolkit, the Data Parallel C++ (DPC++) compatibility tool and profiler, analyzer tools.

Introduction to edgesoftware Command Line Interface (CLI)

edgesoftware is a command line interface (CLI) that helps you manage packages in the Intel's Developer Catalog.

This guide describes the CLI commands and their usage. In this guide you:

- Try out commands and get familiar with CLI and the package you installed
- Learn how to update modules
- Learn how to install custom components
- Learn how to export a package you installed, including custom modules, so that you can install it on other edge nodes.

Get Started with the edgesoftware CLI

1. Open a terminal window.
2. Go to the `edge_insights_for_amr` directory.
3. Try out the following commands.

Get Help or List the Available Commands

- Command:

```
./edgesoftware --help
```

- Response:

```
Usage: edgesoftware [OPTIONS] COMMAND [ARGS]...
A CLI wrapper for management of Intel® Edge Software Hub packages


Options:
  -v, --version   Show the version number and exit.
  --help          Show this message and exit.


Commands:
  download  Download modules of a package.
  export    Exports the modules installed as a part of a package.
  install   Install modules of a package.
  list      List the modules of a package.
  log       Show log of CLI events.
  pull      Pull Docker image.
  uninstall Uninstall the modules of a package.
  update    Update the modules of a package.
  upgrade   Upgrade a package.
```

View the Software Version

- Command:

```
./edgesoftware --version
```

- Response: The edgesoftware version, build date, and target OS.

List the Package Modules

- Command:

```
./edgesoftware list
```

- Response: The modules installed and status.

```
+--------------------------+-----------------------------------+---------+
|            ID            |              Module               | Status  |
+--------------------------+-----------------------------------+---------+
| 605cab935a4b53002c272678 |   Docker Community Edition CE     | SUCCESS |
| 60e58fca4c1e9d002a6d6b2a |          Docker Compose           | SUCCESS |
| 6274c15ae57550002ca63db7 |  amr-aaeon-amr-interface:2022.2   | SUCCESS |
| 6274c1b0e57550002ca63db8 |    amr-battery-bridge:2022.2      | SUCCESS |
| 6274c3b8e57550002ca63dbc |      amr-collab-slam:2022.2       | SUCCESS |
| 6274c402e57550002ca63dbd |      amr-fastmapping:2022.2       | SUCCESS |
| 6274ef41e57550002ca63dd6 |       amr-imu-tools:2022.2        | SUCCESS |
| 6274c680e57550002ca63dc2 |          amr-nav2:2022.2          | SUCCESS |
| 6274c6d2e57550002ca63dc3 |   amr-object-detection:2022.2     | SUCCESS |
| 6274c8f0e57550002ca63dc4 |       amr-realsense:2022.2        | SUCCESS |
| 6274c26be57550002ca63db9 |       amr-ros-base:2022.2         | SUCCESS |
| 6274d29fe57550002ca63dcb |       amr-wandering:2022.2        | SUCCESS |
| 629619006aa8fa002b3af506 |  amr-robot-localization:2022.2    | SUCCESS |
| 6274c0efe57550002ca63db6 |        amr-adbscan:2022.2         | SUCCESS |
| 6274c31ae57550002ca63dbb |      amr-cartographer:2022.2      | SUCCESS |
| 6274c2d1e57550002ca63dba |       amr-fdo-client:2022.2       | SUCCESS |
| 6274c469e57550002ca63dbe |   amr-fleet-management:2022.2     | SUCCESS |
| 6274c5f1e57550002ca63dbf |        amr-gazebo:2022.2          | SUCCESS |
| 6274c62ae57550002ca63dc0 |       amr-gstreamer:2022.2        | SUCCESS |
| 6274c65ae57550002ca63dc1 |        amr-kobuki:2022.2          | SUCCESS |
| 6274c963e57550002ca63dc5 |      amr-ros-arduino:2022.2       | SUCCESS |
| 6274ca11e57550002ca63dc6 |      amr-ros1-bridge:2022.2       | SUCCESS |
| 627cb962e57550002ca63dd9 |     amr-ros2-openvino:2022.2      | SUCCESS |
| 6274cf62e57550002ca63dc7 |        amr-rplidar:2022.2         | SUCCESS |
| 6274cfaae57550002ca63dc8 |        amr-rtabmap:2022.2         | SUCCESS |
| 6287a2a0e57550002ca63de2 |     amr-sick-nanoscan:2022.2      | SUCCESS |
| 6274e5cee57550002ca63dd4 |      amr-slam-toolbox:2022.2      | SUCCESS |
| 6274d110e57550002ca63dc9 |      amr-turtlebot3:2022.2        | SUCCESS |
| 6273f556e57550002ca63db2 |       amr-turtlesim:2022.2        | SUCCESS |
| 6274d163e57550002ca63dca |        amr-vda5050:2022.2         | SUCCESS |
| 6274e848e57550002ca63dd5 |  amr-vda5050-ros2-bridge:2022.2   | SUCCESS |
| 6274dd3be57550002ca63dd1 |     eiforamr-base-sdk:2022.2      | SUCCESS |
| 6274e3a5e57550002ca63dd3 | eiforamr-full-flavour-sdk:2022.2  | SUCCESS |
| 6274dd7be57550002ca63dd2 |   eiforamr-openvino-sdk:2022.2    | SUCCESS |
+--------------------------+-----------------------------------+---------+
```

List Modules Available for Download

- Command:

```
./edgesoftware list --default
```

- Response: All modules available for download for that package version, modules ID and version.

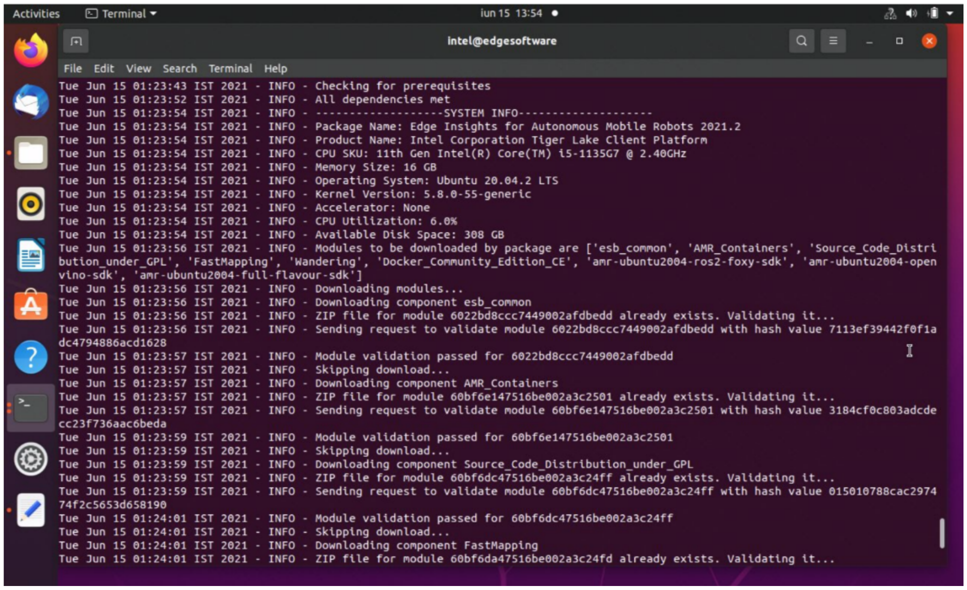
Display the CLI Event Log

- Command:

```
./edgesoftware log
```

- Response: CLI event log information, such as:

- target system information (hardware and software)
- system health
- installation status
- modules you can install



See the Installation Event Log for a Module

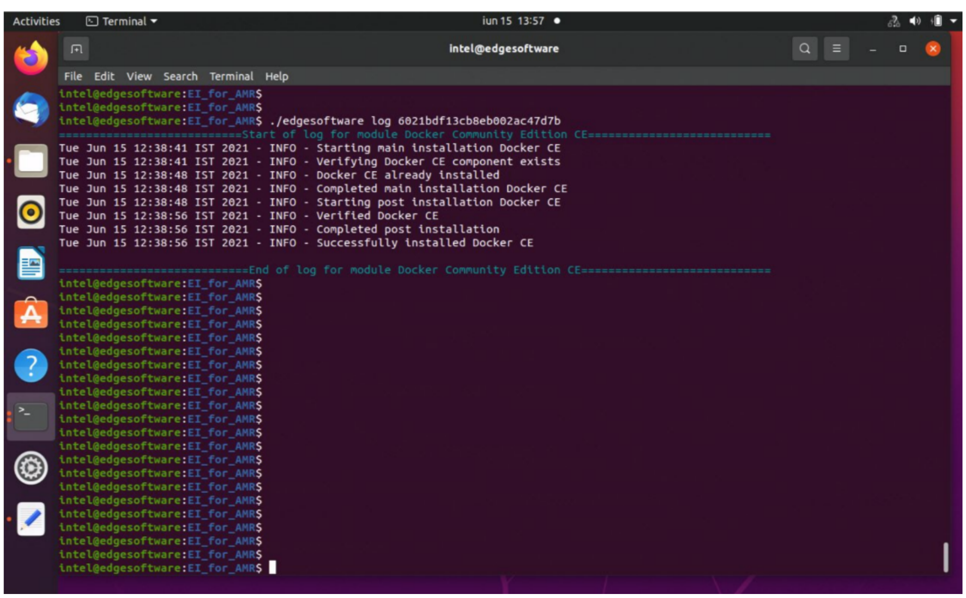
- Command:

```
./edgesoftware log <MODULE_ID>
```

  You can specify multiple `<MODULE_ID>` arguments by listing them with a space between each.

> **NOTE** To find the module ID, use:
>
> ```
> ./edgesoftware list
> ```

- Response: The installation log for the module.



Troubleshooting

If the following error is encountered:

```
PermissionError: [Errno 13] Permission denied: '/var/log/esb-cli/
Edge_Insights_for_Autonomous_Mobile_Robots_2021.3/output.log'
```

Run the CLI commands with sudo:

```
sudo ./edgesoftware  <CLI_commands>
```

Install Package Modules

This edgesoftware command installs package modules on the target system. To do so, the command looks at `edgesoftware_configuration.xml` that was downloaded from the Intel Edge Software Hub when you installed the Edge Insights for Autonomous Mobile Robots software. This file contains information about the modules to install.

During the installation, you are prompted to enter your product key. The product key is in the email message you received from Intel confirming your Edge Insights for Autonomous Mobile Robots download.

Warning Do not manually edit `edgesoftware_configuration.xml`.

1. Open a terminal window.
2. Go to the `edge_insights_for_amr` directory.
3. Run the install command:

```
./edgesoftware install
```

Update the Package Modules

NOTE On a fresh Linux* installation, you might need to use the `install` command at least once before performing an update. `install` makes sure all dependencies and packages are installed on the target system.

```
./edgesoftware install
```

When you are ready to perform the update, use:

```
./edgesoftware update <MODULE_ID>
```

```
Please enter the Product Key. The Product Key is contained in the email you receiv
Updating ['60e58fca4c1e9d002a6d6b2a'] modules of package 627cbc12423d3f002c960952
Starting the setup...
ESB CLI version: 2022.2
Target OS: Ubuntu 20.04
Python version: 3.8.10
Checking Internet connection
Connected to the Internet
Validating package product key
Successfully validated Product Key
Checking for prerequisites
All dependencies met
-------------------SYSTEM INFO--------------------
Package Name: Edge Insights for Autonomous Mobile Robots 2022.2
Product Name: AAEON UPX-TGL01
CPU SKU: 11th Gen Intel(R) Core(TM) i7-1185GRE @ 2.80GHz
Memory Size: 15 GB
Operating System: Ubuntu 20.04.3 LTS
Kernel Version: 5.13.0-1017-intel
Accelerator(VPU): 2
CPU Utilization: 0.5%
Available Disk Space: 133 GB
Starting installation
Downloading modules...
Downloading component Docker_Community_Edition_CE
ZIP file for module 605cab935a4b53002c272678 already exists. Validating it...
Module validation passed for 605cab935a4b53002c272678
Skipping download...
Downloading component Docker_Compose
ZIP file for module 60e58fca4c1e9d002a6d6b2a already exists. Validating it...
Module validation passed for 60e58fca4c1e9d002a6d6b2a
Skipping download...
Downloading modules completed...
Modules to be installed by package are ['Docker_Community_Edition_CE', 'Docker_Com
Docker_Community_Edition_CE is already installed. Type YES to reinstall or NO to s
NO
Docker_Compose is already installed. Type YES to reinstall or NO to skip installat
NO
Installation of package complete
```

During the installation, you are prompted to enter your product key. The product key is in the email message you received from Intel confirming your Edge Insights for Autonomous Mobile Robots download.

NOTE To find the module ID, use:

```
./edgesoftware list -d
```

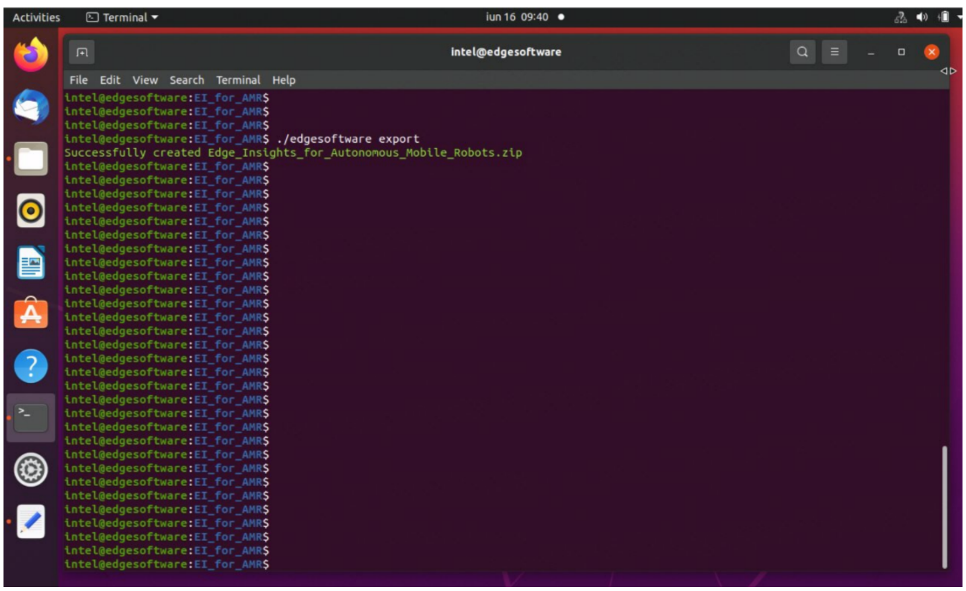
```
| 627bb9ffaf4093002b29b352 |              Source Code GPL              |  2022.2 |
| 627bb982af4093002b29b345 |              AMR Edge Server              |  2022.2 |
| 627bb949af4093002b29b338 |               AMR Containers              |  2022.2 |
| 60e58fca4c1e9d002a6d6b2a |              Docker Compose*              |  1.29.0 |
| 605cab935a4b53002c272678 |      Docker Community Edition (CE)*       | 20.10.5 |
| 62717851af4093002b29a8d8 | Intel® Edge Software Device Qualification |   8.0   |
| 6299a9d518eee6002c250282 |               AMR Bag Files               |  2022.2 |
| 62a0b6a045e006002cc87f35 |              AMR Test Module              |  2022.2 |
+--------------------------+-------------------------------------------+---------+
```

Export the Package for Installation

The edgesoftware CLI lets you package the installed modules, customer applications, and dependencies as part of a package. The export is provided in a `.zip` file that includes installation scripts, XML files, and an edgesoftware Python* executable.

Command:

```
./edgesoftware export
```



Uninstall the Packages

The edgesoftware CLI lets you uninstall the complete package or individual components from the package.

To uninstall an individual package, run the following command:

```
./edgesoftware uninstall <MODULE_ID>
```

To uninstall all the packages, run the following command:

```
./edgesoftware uninstall -a
```

NOTE This command does not uninstall Docker* Compose and Docker* Community Edition (CE).

Edge Insights for Autonomous Mobile Robots Tutorials

These are general tutorials for Edge Insights for Autonomous Mobile Robots.

Robot Tutorials

Follow the tutorials in this section to learn how to use and configure Edge Insights for Autonomous Mobile Robots (EI for AMR).

With step-by-step instructions covering real world usage scenarios, tutorials provide a learning path for developers to follow for mastering the usage of Edge Insights for Autonomous Mobile Robots.

Get started in your learning journey with the Run a ROS 2 Sample Application in the Docker* Container tutorial, and continue with the other tutorials.

You can execute the following sample applications on the `eiforamr-full-flavour-sdk` container.

- Run a ROS 2 Sample Application in the Docker* Container
- Run an Intel® RealSense™ ROS 2 Sample Application in Docker* Container
- Run a GStreamer* Video Pipeline using GStreamer* Plugins in Docker* Container
- Run a GStreamer* Audio Pipeline using GStreamer* Plugins in Docker* Container
- Run a GStreamer* Video Pipeline using libv4l2 in Docker* Container
- Run a GStreamer* Video Pipeline using the Intel® RealSense™ Plugin in the Docker* Container
- Optional (Only for Sony* IMX390 Setups): Run a GStreamer* Video Pipeline using Sony's IMX390 MIPI Sensor in Docker* Container
- Run OpenVINO™ Sample Applications in Docker* Container
- Run ROS 2 OpenVINO™ Toolkit Sample Applications in Docker* Container
- Run Intel® oneAPI Base Toolkit Sample Applications in Docker* Container
- Run Profiling Application in Docker* Container with VTune™ Profiler
- Run OpenVINO™ Benchmarking Tool
- Run the ADBSCAN Algorithm
- Launch Wandering Application with Gazebo* Simulation
- Launch Wandering Application on AAEON* Robot Kit
- Launch Cartographer with 2D LIDAR
- Run FastMapping Algorithm
- Run ROS 2 Navigation Sample Applications Using the ITS Path Planner Plugin in a Docker* Container
- ITS Path Planner Plugin Customization
- Run the Edge Insights for Autonomous Mobile Robots Container in KVM Guest
- Run a Collaborative SLAM System
- Build New and Custom Docker* Images from the Edge Insights for Autonomous Mobile Robots SDK
- Troubleshooting for Robot Tutorials

Run a ROS 2 Sample Application in the Docker* Container

1. Go to the `AMR_containers` folder:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_*/AMR_containers
```

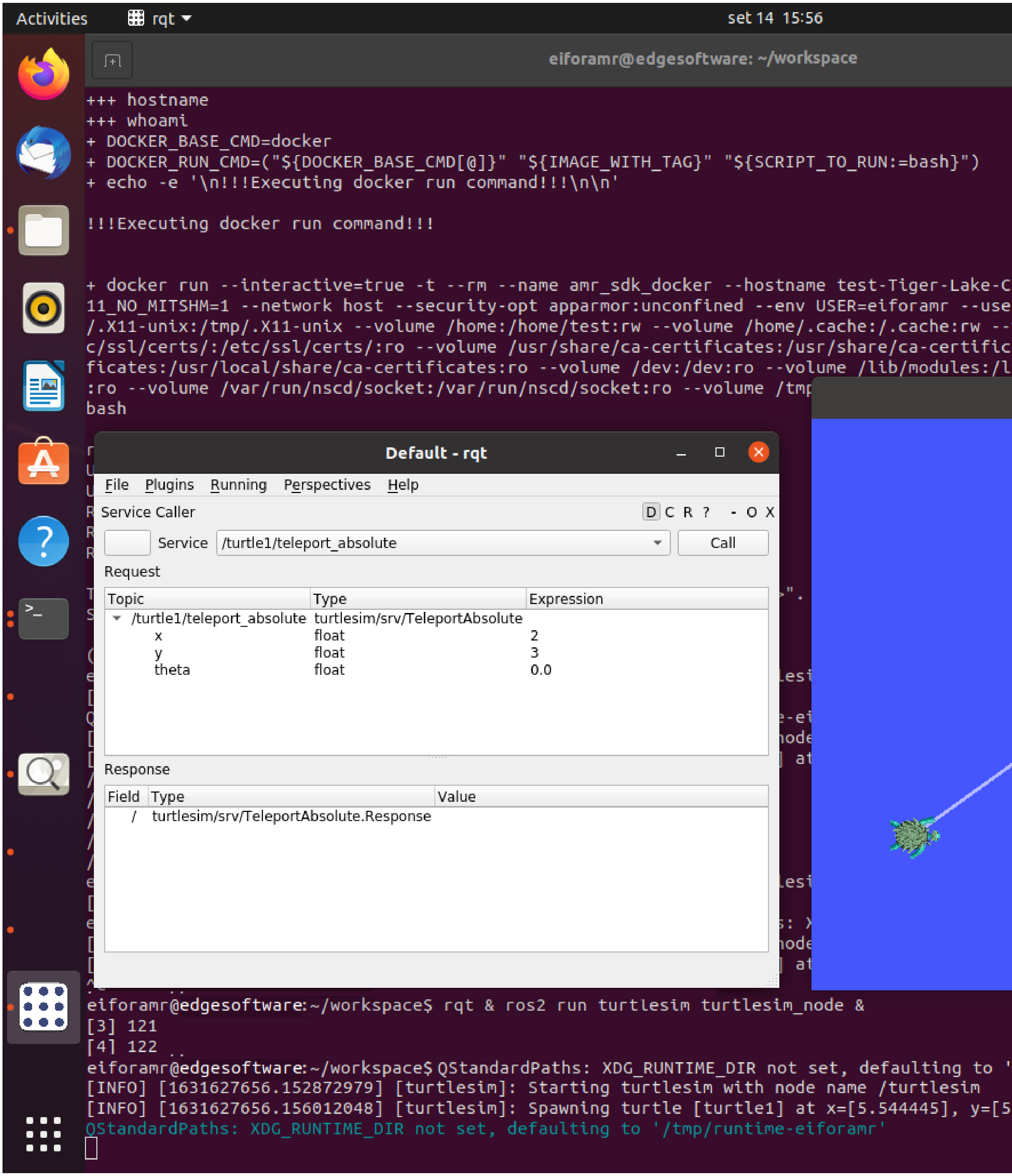
2. Prepare the environment setup:

```
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
export CONTAINER_BASE_PATH=`pwd`
export ROS_DOMAIN_ID=12
```

3. Run an automated yml file that opens a ROS 2 sample application inside the EI for AMR Docker* container.

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
turtlesim.demo.yml up
```

4. Go to **Plugins > Services > Service Caller**: Choose to move turtle1 by choosing (from the Service drop-down list) /turtle1/teleport_absolute and make sure you changed x and y coordinates for the original values. Press **Call**. The turtle should move. Close the service caller window by pressing **x**. Then type `Ctrl-c`.



5. To close this, do one of the following:

- Type `Ctrl-c` in the terminal where you did the up command.
- Close the rqt window.

- Run this command in another terminal:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
turtlesim.demo.yml down
```

6. For an explanation of what happened, open the yml file:
   - The first 23 lines are from the EI for AMR infrastructure.
   - Line 26 starts the `turtlesim` ROS 2 node.
   - Line 31 starts the rqt so that the turtle can be controlled.

Run an Intel® RealSense™ ROS 2 Sample Application in Docker* Container

This tutorial tells you how to:

- Launch ROS nodes for a camera.
- List ROS topics.
- See that Intel® RealSense™ topics are publishing data.
- Get data from the Intel® RealSense™ camera (data coming at FPS).
- See an image from the Intel® RealSense™ camera displayed in rviz2.

Run the Sample Application

1. Connect an Intel® RealSense™ camera (for example, D435i) to the host.
2. Go to the `AMR_containers` folder:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_*/AMR_containers
```

3. Prepare environment setup:

```
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
export CONTAINER_BASE_PATH=`pwd`
export ROS_DOMAIN_ID=12
```

4. Run the command below to start the Docker container:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/01_amr/amr-sdk.all.yml
run realsense bash
```

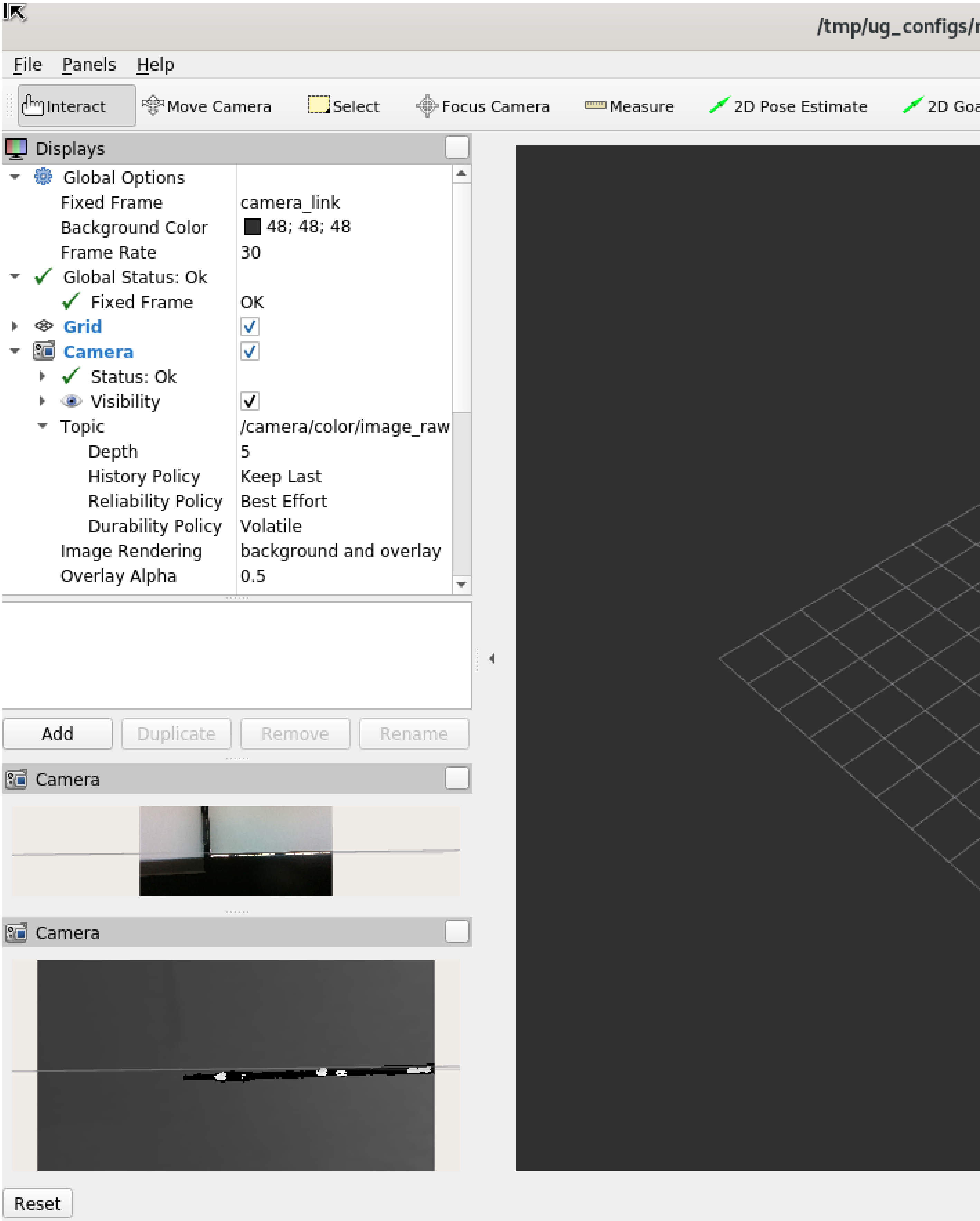
5. Check for latest Intel® RealSense™ firmware updates.
   a. Open the Intel® RealSense™ viewer application:

```
realsense-viewer
```

   In the Intel® RealSense™ viewer, if any firmware update is available, a window popup appears in the upper right corner.
   b. During the firmware update installation, do not disconnect the Intel® RealSense™ camera. Press **Install** in the window popup.
   c. After the installation is complete or if no update is available, close the Intel® RealSense™ viewer.
6. Run an automated yml file that opesn the Intel® RealSense™ ros2 node and lists camera-relevant information.

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
realsense.demo.yml up
```

   Expected output: The image from the Intel® RealSense™ camera is displayed in rviz2, on the bottom left side.


/tmp/ug_configs/r
File
Panels
Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Goa
Displays
Global Options
Fixed Frame camera_link
Background Color 48; 48; 48
Frame Rate 30
Global Status: Ok
Fixed Frame OK
Grid
Camera
Status: Ok
Visibility
Topic /camera/color/image_raw
Depth 5
History Policy Keep Last
Reliability Policy Best Effort
Durability Policy Volatile
Image Rendering background and overlay
Overlay Alpha 0.5
Add
Duplicate
Remove
Rename
Camera
Camera
Reset

7. To close this, do one of the following:
   - Type `Ctrl-c` in the terminal where you did the up command.
   - Run this command in another terminal:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
realsense.demo.yml down
```

Troubleshooting

In some cases, the stream may not appear due to permission issues on the host. You may see this error message:

```
ERROR: Pipeline doesn't want to pause.
```

1. To fix this, install the librealsense udev rules.

```
git clone https://github.com/IntelRealSense/librealsense
Copy the 99-realsense-libusb.rules files to the rules.d folder
sudo cp config/99-realsense-libusb.rules /etc/udev/rules.d/
sudo udevadm control --reload-rules
sudo udevadm trigger
```

2. Then open the `gst-launch` command.

If the problem persists, you can try any or all of the following:

- Verify that $DISPLAY has the correct value.
- Perform an Intel® RealSense™ hardware reset:

```
# Open realsense docker container
docker-compose -f 01_docker_sdk_env/docker_compose/01_amr/amr-sdk.all.yml run realsense bash
# While in realsense container, open the realsense-viewer application
realsense-viewer
# In  realsense-viewer menu, go to "More" and then select "Hardware Reset"
# Wait for reset to complete and then close the realsense-viewer application.
```

- Reboot the target.

For Intel® RealSense™ documentation, see https://dev.intelrealsense.com/docs/docs-get-started.

For calibration issues, see https://dev.intelrealsense.com/docs/self-calibration-for-depth-cameras.

For general robot issues, go to: Troubleshooting for Robot Tutorials.

Run a GStreamer* Video Pipeline using GStreamer* Plugins in Docker* Container

This tutorial tells you how to run a GStreamer* video pipeline using GStreamer* plugins and display a video file in a window in the container.

Run the Sample Application

1. Go to the `AMR_containers` folder:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_*/AMR_containers
```

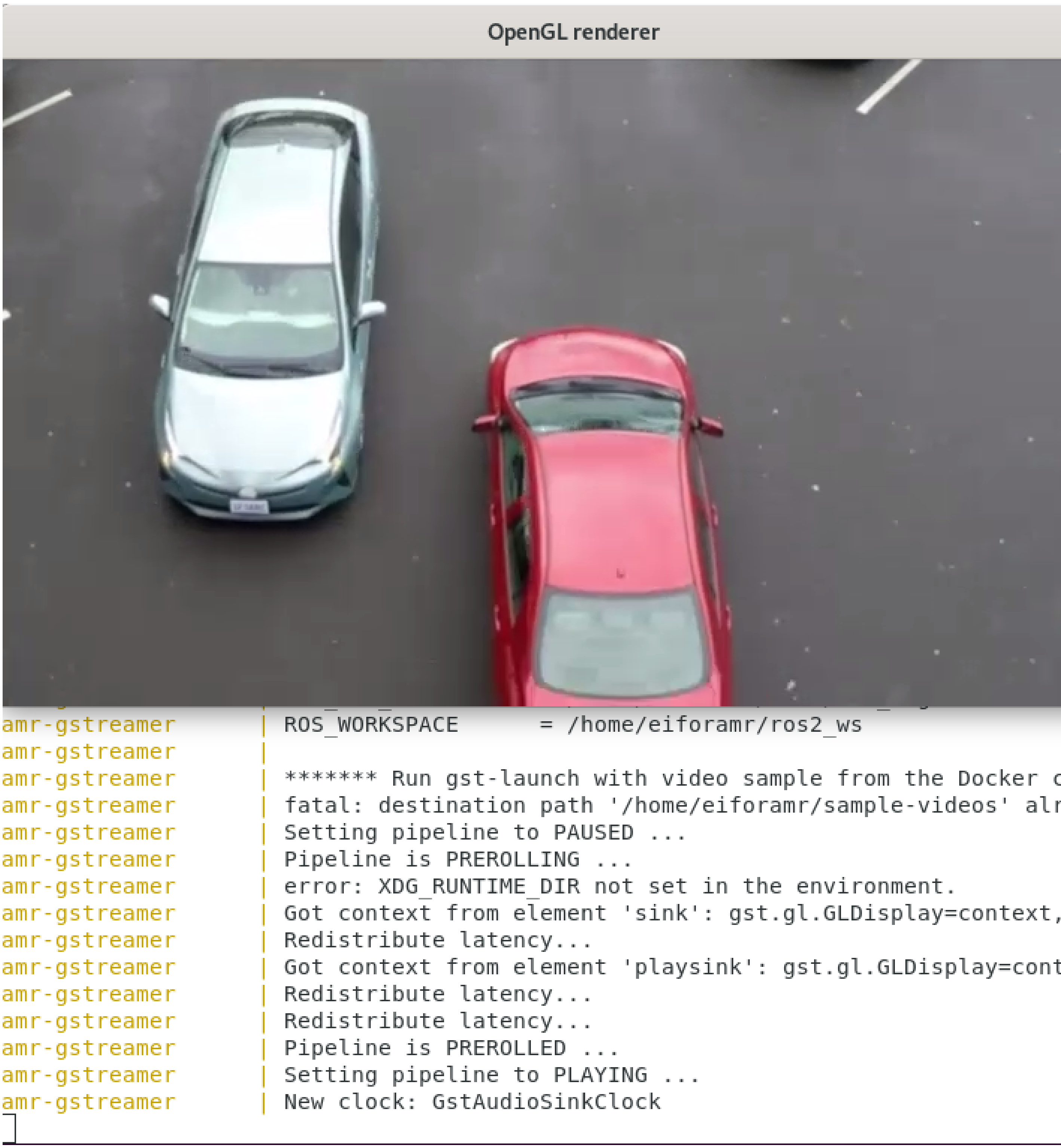
2. Prepare environment setup:

```
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
export CONTAINER_BASE_PATH=`pwd`
export ROS_DOMAIN_ID=31
```

3. Run an automated yml file that opens a GStreamer* sample application inside the EI for AMR Docker* container.

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
gstreamer_video.demo.yml up
```

Expected output: The video file is displayed in a window in the container.



4. To close this, do one of the following:

   - Type `Ctrl-c` in the terminal where you did the up command.
   - Run this command in another terminal:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
gstreamer_video.demo.yml down
```

5. For an explanation of what happened, open the yml file:

   - The first 23 lines are from the EI for AMR infrastructure.
   - Line 26 clones some sample videos.
   - Line 27 play the video using GStreamer*.

6. To use your own video, use the same yml file but update line 27 to target your own file.

   For example, copy the file:

```
cp test.mp4 ${CONTAINER_BASE_PATH}/01_docker_sdk_env/docker_compose/05_tutorials/test.mp4
```

   And update line 27 to:

```
gst-launch-1.0 playbin uri=file://${CONTAINER_BASE_PATH}/01_docker_sdk_env/docker_compose/
05_tutorials/test.mp4
```

Troubleshooting

Check your system date and time:

```
date
```

If the date is incorrect, contact your local support team for help setting the correct date and time.

For general robot issues, go to: Troubleshooting for Robot Tutorials.

Run a GStreamer* Audio Pipeline using GStreamer* Plugins in Docker* Container

Run the Sample Application

1. Go to the `AMR_containers` folder:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_containers
```

2. Prepare environment setup:

```
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
export CONTAINER_BASE_PATH=`pwd`
export ROS_DOMAIN_ID=34
```

3. Run an automated yml file that opens a GStreamer* sample application inside the EI for AMR Docker* container.

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
gstreamer_audio.demo.yml up
```

   Expected output:

```
#gst-launch-1.0 filesrc location=/data_samples/media_samples/sample.ogg ! oggdemux ! vorbisdec !
audioconvert ! audioresample ! Testsink
error: XDG_RUNTIME_DIR not set in the environment.
Setting pipeline to PAUSED ...
Pipeline is PREROLLING ...
Pipeline is PREROLLED ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock
Got EOS from element "pipeline0".
```

```
Execution ended after 0:01:14.349609320
Setting pipeline to PAUSED ...
Setting pipeline to READY ...
Setting pipeline to NULL ...
Freeing pipeline ...
```

4. To close this, do one of the following:

    - Type `Ctrl-c` in the terminal where you did the up command.
    - Run this command in another terminal:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
gstreamer_audio.demo.yml down
```

5. For an explanation of what happened, open the yml file:

    - The first 23 lines are from the EI for AMR infrastructure.
    - Line 26 plays the audio file using GStreamer*.

6. To use your own audio, use the same yml file but update line 26 to target your own file.

    For example, copy the file:

```
cp test.ogg ${CONTAINER_BASE_PATH}/01_docker_sdk_env/docker_compose/05_tutorials/test.ogg
```

    And update line 26 to:

```
gst-launch-1.0 filesrc location=${CONTAINER_BASE_PATH}/01_docker_sdk_env/docker_compose/
05_tutorials/test.ogg ! oggdemux ! vorbisdec ! audioconvert ! audioresample ! testsink
```

Troubleshooting

For general robot issues, go to: Troubleshooting for Robot Tutorials.

Run a GStreamer* Video Pipeline using libv4l2 in Docker* Container

Run the Sample Pipeline

1. Connect a video camera compatible with libv4l2, such as a webcam (an Intel® RealSense™ camera is not compatible).
2. Go to the `AMR_containers` folder:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_containers
```

3. Prepare environment setup:

```
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
export CONTAINER_BASE_PATH=`pwd`
export ROS_DOMAIN_ID=27
sudo chmod a+rw /dev/video*
```

4. Get the stream from the webcam using GStreamer*:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
gstreamer_libv4l2.demo.yml up
```

    Expected output: The stream from the webcam is displayed.

5. To close this, do one of the following:

    - Type `Ctrl-c` in the terminal where you did the up command.
    - Run this command in another terminal:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
gstreamer_libv4l2.demo.yml down
```

6. For an explanation of what happened, open the yml file:

- The first 23 lines are from the EI for AMR infrastructure.
- Line 26 plays the stream from the webcam using GStreamer*.

Troubleshooting

If the following error is encountered:

```
eiforamr@edgesoftware:~/workspace$ gst-launch-1.0 v4l2src ! autovideosink
Setting pipeline to PAUSED ...
Pipeline is live and does not need PREROLL ...
Setting pipeline to PLAYING ...
ERROR: from element /GstPipeline:pipeline0/GstV4l2Src:v4l2src0: Internal data stream error.
Additional debug info:
gstbasesrc.c(3072): gst_base_src_loop (): /GstPipeline:pipeline0/GstV4l2Src:v4l2src0:
streaming stopped, reason not-negotiated (-4)
Execution ended after 0:00:00.000028689
Setting pipeline to PAUSED ...
Setting pipeline to READY ...
Setting pipeline to NULL ...
Freeing pipeline ...
```

GStreamer* may want the type of decoding added. For example, for a Logitech* C922 webcam, the command is:

```
$ gst-launch-1.0 v4l2src ! jpegdec ! autovideosink
```

If the following error is encountered:

```
amr-gstreamer | Setting pipeline to PAUSED ...
amr-gstreamer | error: XDG_RUNTIME_DIR not set in the environment.
```

Try this:

```
mkdir -pv ~/.cache/xdgr
export XDG_RUNTIME_DIR=$PATH:~/.cache/xdgr
```

For general robot issues, go to: Troubleshooting for Robot Tutorials.

Run a GStreamer* Video Pipeline using the Intel® RealSense™ Plugin in the Docker* Container

This tutorial tells you how to run a GStreamer* video pipeline using a Intel® RealSense™ video camera as the video source.

Run the Sample Pipeline

1. Connect an Intel® RealSense™ video camera.
2. Go to the `AMR_containers` folder:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_containers
```

3. Prepare environment setup:

```
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
export CONTAINER_BASE_PATH=`pwd`
export ROS_DOMAIN_ID=45
sudo chmod a+rw /dev/video*
```

4. Get the stream from the Intel® RealSense™ camera using gstreamer:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
gstreamer_realsensesrc.demo.yml up
```

   Expected output: The stream from the Intel® RealSense™ is displayed.

5. To close this, do one of the following:
    - Type `Ctrl-c` in the terminal where you did the up command.
    - Run this command in another terminal:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
gstreamer_realsensesrc.demo.yml down
```

6. For an explanation of what happened, open the yml file:
    - The first 23 lines are from the EI for AMR infrastructure.
    - Line 26 gets the stream from the Intel® RealSense™ app using GStreamer*.

Troubleshooting

- In some cases, the stream may not appear due to permission issues on the host. You may see this error message:

```
ERROR: Pipeline doesn't want to pause.
```

  1. To fix this, install the librealsense udev rules:

```
git clone https://github.com/IntelRealSense/librealsense
Copy the 99-realsense-libusb.rules files to the rules.d folder
sudo cp config/99-realsense-libusb.rules /etc/udev/rules.d/
sudo udevadm control --reload-rules
sudo udevadm trigger
```

  2. Then open the `gst-launch` command.

  If the problem persists, you can try any or all of the following:

  - Verify that $DISPLAY has the correct value.
  - Perform a Intel® RealSense™ hardware reset:

```
# Open realsense docker container
docker-compose -f 01_docker_sdk_env/docker_compose/01_amr/amr-sdk.all.yml run realsense bash
# While in realsense container, open the realsense-viewer application
realsense-viewer
# In  realsense-viewer menu, go to "More" and then select "Hardware Reset"
# Wait for reset to complete and then close the realsense-viewer application.
```

  - Reboot the target.
- For general robot issues, go to: Troubleshooting for Robot Tutorials.

Optional (Only for Sony* IMX390 Setups): Run a GStreamer* Video Pipeline using Sony's IMX390 MIPI Sensor in Docker* Container

This tutorial tells you how to set up and run a GStreamer* video pipeline using Sony's IMX390 MIPI sensor.

Prerequisites: To enable Sony's IMX390 MIPI sensor, you must use the Resource Design Center (RDC), have a Corporate Non-Disclosure Agreement (CNDA) in place, and ask for download access.

Step 1: Prepare the Target System

Make sure your target system has a fresh installation of Ubuntu* Linux* that corresponds to the version of Edge Insights for Autonomous Mobile Robots (EI for AMR) that you downloaded. If you need help installing Ubuntu*, follow these steps:

1. Download Ubuntu* Linux* Desktop ISO file to your developer workstation.
2. Create a bootable flash drive using an imaging application, such as Startup Disk Creator, available on Ubuntu*.
3. After flashing the USB drive, power off your target system, insert the USB drive, and power on the target system.

If the target system does not boot from the USB drive, change the boot priority in the system BIOS.

4. Follow the prompts to install Ubuntu* with the default configurations. For detailed instructions, see the Ubuntu* guide.

> **NOTE** When creating your login details, do not use `eiforamr` as a username because it is used in the Automated Mobile Robots Docker images. If the system has this username, it will crash when it tries to open AMR Docker images using this command: `./run_interactive_docker.sh eiforamr-full-flavour-sdk:<TAG> eiforamr`

5. Power down your target system, and remove the USB drive.
6. Power up the target system.

   Expected result: You see that Ubuntu* Desktop is successfully installed.

Step 2: Update the Kernel to Version 5.10.109

The only supported kernel version for EI for AMR is Intel's Linux* LTS Kernel 5.10.109. Depending on your Ubuntu* 20.04 version, the default kernels are:

- 5.4 on Ubuntu* 20.04.1
- 5.8 on Ubuntu* 20.04.2
- 5.11 on Ubuntu* 20.04.3
- 5.13 on Ubuntu* 20.04.4

To check your kernel version:

```
uname -r
```

Step 2 is only valid for updating from kernel versions 5.4, 5.8, 5.11 and 5.13. If you have a different kernel, go to the Support Forum.

This process can take from 30 minutes to two hours, depending on your system.

1. Clone Intel's Linux* LTS kernel 5.10.109 repository from GitHub*.

```
git clone https://github.com/intel/linux-intel-lts.git
cd linux-intel-lts
```

2. Check out to the `lts-v5.10.109-yocto-220512T050604Z` branch.

```
git checkout lts-v5.10.109-yocto-220512T050604Z
```

3. Install the necessary packages and `gcc` dependencies.

```
sudo apt-get -y install build-essential gcc bc bison flex libssl-dev libncurses5-dev libelf-dev
dwarves zstd
```

4. Copy the configuration file to your folder, and rename it `.config`.

```
cp /boot/config-$(uname -r) ./.config
```

5. Change these values of the kernel configuration.

```
scripts/config --set-str SYSTEM_TRUSTED_KEYS ""
scripts/config --set-str CONFIG_SYSTEM_REVOCATION_KEYS ""
```

6. Enable the Sony* IMX390-related and TI* TI960-related modules.

```
scripts/config --module CONFIG_VIDEO_IMX390
scripts/config --module CONFIG_VIDEO_TI960
scripts/config --module CONFIG_VIDEO_INTEL_IPU6
scripts/config --module CONFIG_VIDEO_AR0234
scripts/config --module CONFIG_PINCTRL_TIGERLAKE
scripts/config --enable CONFIG_INTEL_IPU6_TGLRVP_PDATA
```

7. Compile the kernel, and make the Debian* kernel packages.

NOTE

This kernel compilation step takes a long time to complete:

- approximately one hour on systems with 32 GB of RAM
- two to three hours on systems with 8 GB of RAM

```
make olddefconfig
make -j4 deb-pkg
```

8. Install the new Debian* kernel packages.

```
cd ../ && sudo dpkg -i linux-*.deb
```

9. For **kernel versions 5.11 and 5.13**, the newly installed kernel is a lower version than the system kernel, so the system needs to be configured to use it instead of the latest version.

a. Open the GRUB.

```
sudo cp /etc/default/grub /etc/default/grub.bak
sudo vi /etc/default/grub
```

b. Change the value of `GRUB_DEFAULT` from `GRUB_DEFAULT=0` to `GRUB_DEFAULT="Advanced options for Ubuntu>Ubuntu, with Linux 5.10.109"`.

c. Update the GRUB.

```
cd /tmp
sudo update-grub
```

10. Reboot your system.

```
sync
sudo reboot -fn
```

11. Check your kernel version after reboot.

```
uname -r
```

Step 3: Install the IPU6 Packages

1. Download the Tiger Lake IPU6 Packages on the host (it contains the IPU RPM libraries and IPU firmware).

2. Unzip the archive, and copy the RPM folder to the /tmp folder:

```
unzip 645460.zip
tar -xf ipu6_rpm_beta.tar.bz2
cp -r rpm /tmp
```

3. Run the Docker* image as root:

```
xhost +
./run_interactive_docker.sh  amr-gstreamer:<TAG> root -e "--volume /sys/kernel/:/sys/kernel:rw --
volume /sys/class:/sys/class:rw"
```

4. If your network runs behind proxies, export the corresponding proxies in the container.

```
export http_proxy="http://<http_proxy>:port"
export https_proxy="http://<https_proxy>:port"
```

5. Install the RPM package in the Docker* container:

```
apt-get update
apt-get install rpm
```

6. Set isys_freq:

```
echo 400 > /sys/kernel/debug/intel-ipu/buttress/isys_freq
```

7. Install the IPU6 firmware and other necessary user-space libraries:

```
rpm -ivh /tmp/rpm/* --nodeps --force
```

8. Prepare the setup:

```
export DISPLAY=:0 #If you are on VNC adapt this value to the correct one
export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig:/usr/lib64/pkgconfig:/usr/lib/pkgconfig
export LD_LIBRARY_PATH=/usr/local/lib:/usr/lib64:/usr/lib
export GST_PLUGIN_PATH=/usr/lib/gstreamer-1.0
export GST_GL_PLATFORM=egl
```

Step 4: Run the Sample Application

1. Run the GStreamer* pipeline:

```
gst-launch-1.0 icamerasrc device-name=imx390 printfps=true num-vc=1 ! video/x-
raw,format=NV12,width=1920,height=1200 ! videoconvert ! xvimagesink
```

Expected output: A video opens, showing images captured with the camera using Sony's IMX390 MIPI sensor.

Troubleshooting

For general robot issues, go to: Troubleshooting for Robot Tutorials.

Run OpenVINO™ Sample Applications in Docker* Container

This tutorial tells you how to:

- Run inference engine object detection on a pretrained network using the SSD method.
- Run the detection demo application for a CPU and GPU.
- Use a model optimizer to convert a TensorFlow* neural network model.
- After conversion, run the neural network with inference engine for a CPU and GPU.

Run the Sample Application

1. Go to the `AMR_containers` folder:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_containers
```

2. Prepare the environment setup:

```
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
export CONTAINER_BASE_PATH=`pwd`
export ROS_DOMAIN_ID=22
```

3. Run inference engine object detection on a pre-trained network using the Single-Shot multibox Detection (SSD) method. Run the detection demo application for a CPU:

```
CHOOSE_USER=root docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
openvino_CPU.demo.yml up
```

Expected output: A video in a loop with cars being detected and labeled by the Neural Network using a CPU

4. To close this, do one of the following:
    - Type `Ctrl-c` in the terminal where you did the up command.
    - Run this command in another terminal:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
openvino_CPU.demo.yml down
```

5. For an explanation of what happened, open the yml file. The file is well documented. To use your own files, place them in your home directory, and change the respective lines in the yml files to target them.
6. Run the detection demo application for the GPU:

```
CHOOSE_USER=root docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
openvino_GPU.demo.yml up
```

Expected output: A video in a loop with cars being detected and labeled by the Neural Network using a GPU

7. To close this, do one of the following:
   - Type `Ctrl-c` in the terminal where you did the up command.
   - Run this command in another terminal:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
openvino_GPU.demo.yml down
```

8. For an explanation of what happened, open the yml file. The file is well documented. To use your own files, place them in your home directory, and change the respective lines in the yml files to target them.
9. For system with an Intel® Movidius™ Myriad™ X accelerator, run the detection demo application on the Intel® Movidius™ Myriad™ X accelerator:

NOTE Only execute this command on systems with an Intel® Movidius™ Myriad™ X accelerator. Check your system:

```
lsusb
```

Look for `Intel Movidius MyriadX` in the output.

```
CHOOSE_USER=root docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
openvino_MYRIAD.demo.yml up
```

Expected output: A video in a loop with cars being detected and labeled by the Neural Network using the Intel® Movidius™ Myriad™ X accelerator.

NOTE There is a known issue that if you choose to run the `object_detection_demo` using the `-d MYRIAD` option, a core dump error is thrown when the demo ends.
If errors occur, remove the following file and try again:

```
rm -rf /tmp/mvnc.mutex
```

10. To close this, do one of the following:
    - Type `Ctrl-c` in the terminal where you did the up command.
    - Run this command in another terminal:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
openvino_MYRIAD.demo.yml down
```

11. For an explanation of what happened, open the yml file. The file is well documented. To use your own files, place them in your home directory, and change the respective lines in the yml files to target them.

Troubleshooting

For general robot issues, go to: Troubleshooting for Robot Tutorials.

Run ROS 2 OpenVINO™ Toolkit Sample Applications in Docker* Container

This tutorial tells you how to run the segmentation demo application on both a static image and on a video stream received from a Intel® RealSense™ camera.

Run the Sample Application

1. Go to the `AMR_containers` folder:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_*/AMR_containers
```

2. Prepare the environment setup:

```
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
export CONTAINER_BASE_PATH=`pwd`
export ROS_DOMAIN_ID=16
```

3. Launch the automated execution of the ROS 2 OpenVINO™ toolkit sample applications:

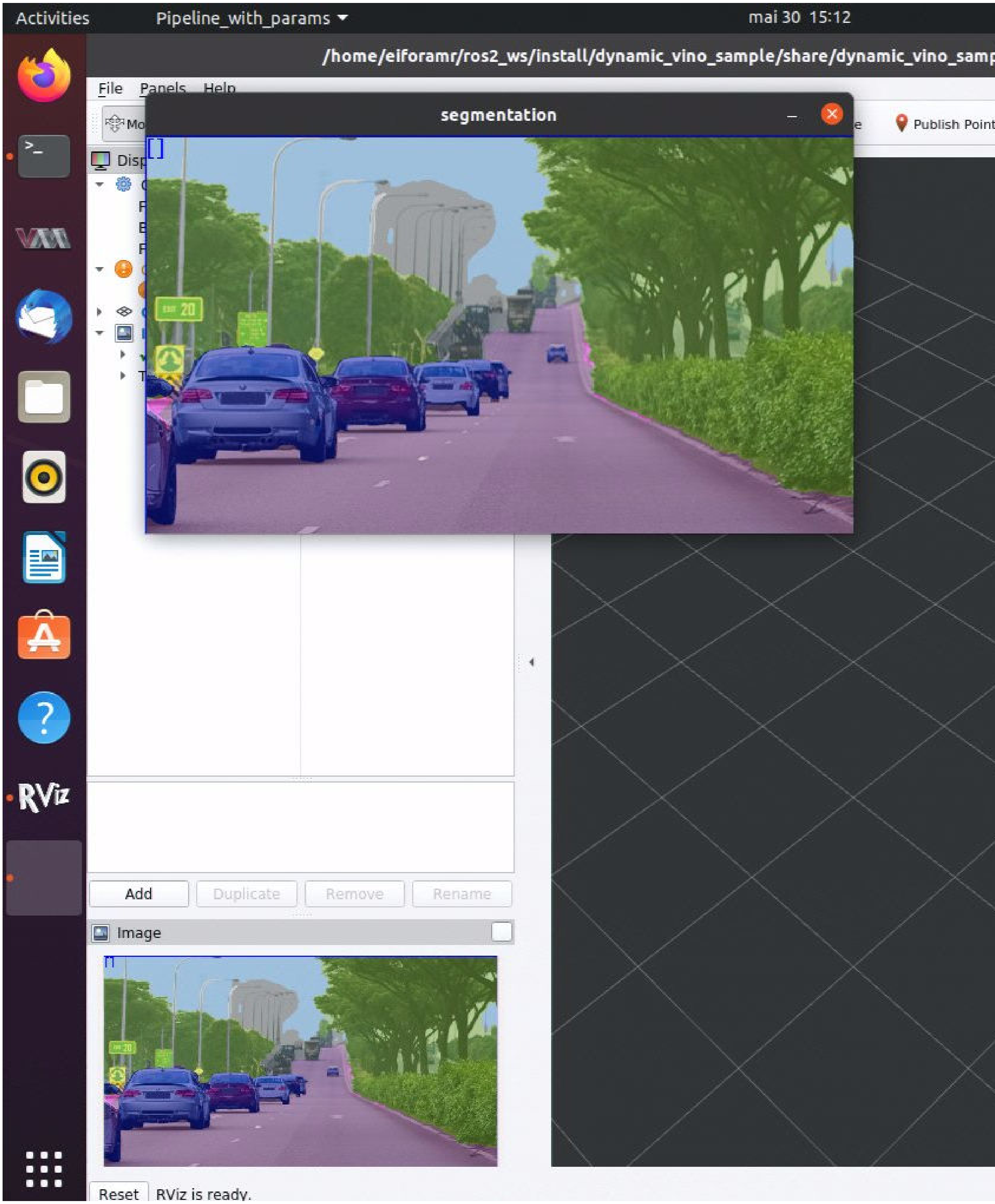
```
CHOOSE_USER=root docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
ros2_openvino.tutorial.yml up
```

   Expected output:

   a. Execution of the object segmentation sample code input from the image: This takes one minute, and you can see the semantic segmentation being applied to the image.

      Original image

Image with semantic object segmentation


Activities
Pipeline_with_params
mai 30 15:12
/home/eiforamr/ros2_ws/install/dynamic_vino_sample/share/dynamic_vino_samp
File Panels Help
segmentation
Publish Point
Add
Duplicate
Remove
Rename
Image
Reset
RViz is ready.

b. Execution of the object segmentation sample code input from the Intel® RealSense™ camera topic: This requires a Intel® RealSense™ camera connected to the testing target. It takes one minute, and you can see the semantic segmentation being applied to the video stream received from a Intel® RealSense™ camera.

4. To close this, do one of the following:
   - Type `Ctrl-c` in the terminal where you did the up command.
   - Run this command in another terminal:

```
CHOOSE_USER=eiforamr 01_docker_sdk_env/docker_compose/05_tutorials/ros2_openvino.tutorial.yml
down
```

How it Works

All of the commands required to run this tutorial are documented in:

```
01_docker_sdk_env/docker_compose/05_tutorials/ros2_openvino.tutorial.yml
```

To use your own image to run semantic segmentation:

1. Copy your image into the `AMR_containers` folder at:

```
cp <path_to_image>/my_image.jpg 01_docker_sdk_env/docker_compose/05_tutorial/param/
```

2. Edit `01_docker_sdk_env/docker_compose/05_tutorials/ros2_openvino.tutorial.yml`, at line 34, adding the following command:

```
cp ${CONTAINER_BASE_PATH}/01_docker_sdk_env/docker_compose/05_tutorials/param/my_image.jpg ../
ros2_ws/src/ros2_openvino_toolkit/data/images/
```

3. Edit `01_docker_sdk_env/docker_compose/05_tutorials/param/pipeline_segmentation_image.yaml` to change the `input_path:`, line 4:

```
input_path: /home/eiforamr/ros2_ws/src/ros2_openvino_toolkit/data/images/my_image.jpg
```

4. Run the automated yml:

```
CHOOSE_USER=root docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
ros2_openvino.tutorial.yml up
```

Expected result: Execution of semantic segmentation on the image you selected

Troubleshooting

For general robot issues, go to: Troubleshooting for Robot Tutorials.

Run Intel® oneAPI Base Toolkit Sample Applications in Docker* Container

This tutorial tells you how to use the DPC++ compiler, convert CUDA to DPC++, build it, and run it.

Run the Sample Application

1. Go to the `AMR_containers` folder:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_containers
```

2. Prepare environment setup:

```
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
```

3. Run the command below to start the Docker container as root:

```
CHOOSE_USER=root docker-compose -f 01_docker_sdk_env/docker_compose/01_amr/amr-sdk.all.yml run
full-sdk bash
```

NOTE If a proxy is required to connect to the Internet, update `/etc/apt/apt.conf.d/proxy.conf` with the corresponding exports and execute the following export commands:

```
echo 'Acquire::http::proxy "<http_proxy:port>";' | sudo tee -a /etc/apt/apt.conf.d/
proxy.conf
echo 'Acquire::https::proxy "<https_proxy:port>";' | sudo tee -a /etc/apt/apt.conf.d/
proxy.conf
export http_proxy="http://<http_proxy>:port"
export https_proxy="http://<https_proxy>:port"
```

4. Install CUDA (replace `<http_proxy:port>` with your proxy):

```
# Send proxy exports
wget -O /etc/apt/preferences.d/cuda-repository-pin-600 https://developer.download.nvidia.com/
compute/cuda/repos/ubuntu1804/x86_64/cuda-ubuntu1804.pin
[ -z "$HTTP_PROXY" ] || apt-key adv --keyserver-options http-proxy=<http_proxy:port> --fetch-
keys https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/3bf863cc.pub
add-apt-repository "deb https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/
x86_64/ /"
apt-get update -y --allow-unauthenticated  && DEBIAN_FRONTEND=noninteractive
sudo apt-get install -y --no-install-recommends cuda-10-1
rm -rf /var/lib/apt/lists/*
```

5. The install command may fail to check if CUDA was installed:

```
ls /usr/local/cuda*
```

Example output:

```
/usr/local/cuda:
LICENSE  README  bin  doc  extras  include  lib64  libnsight  libnvvp  nsightee_plugins  nvml
nvvm  samples  share  src  targets  tools  version.txt

/usr/local/cuda-10.1:
LICENSE  README  bin  doc  extras  include  lib64  libnsight  libnvvp  nsightee_plugins  nvml
nvvm  samples  share  src  targets  tools  version.txt

/usr/local/cuda-10.2:
include  lib64  targets
```

6. Set up the environment for Intel® oneAPI Base Toolkit:

```
source /opt/intel/oneapi/setvars.sh
```

7. Download a sample file that uses the DPC++ compiler:

```
wget -L https://raw.githubusercontent.com/intel/llvm/98b6ee437ed325992ace95548b0ffc01dd4cbbe9/
sycl/examples/simple-dpcpp-app.cpp -O simple.cpp
```

Run the command below and review the output binary:

```
dpcpp simple.cpp -o simple
./simple
```

Expected output:

```
"The results are correct":
```

8. Convert CUDA to DPC++ and build it.

a. Go to CUDA code sample and convert to DPC++:

```
git clone https://github.com/oneapi-src/oneAPI-samples.git
cp oneAPI-samples/Tools/Migration/vector-add-dpct/src/vector_add.cu /home/eiforamr/data_samples/
vector_add.cu
chmod +x /home/eiforamr/data_samples/vector_add.cu
dpct --in-root=/home/eiforamr/data_samples/ /home/eiforamr/data_samples/vector_add.cu
```

Expected output:

```
root@edgesoftware:/home/eiforamr/data_samples# chmod +x "${DATA_SAMPLES}"/vector_add.cu
root@edgesoftware:/home/eiforamr/data_samples# dpct --in-root=/home/eiforamr/data_samples/
vector_add.cu
NOTE: Could not auto-detect compilation database for file 'vector_add.cu' in '/home/eiforamr/
data_samples' or any parent directory.
The directory "dpct_output" is used as "out-root"
Processing: /home/eiforamr/data_samples/vector_add.cu
/home/eiforamr/data_samples/vector_add.cu:32:14: warning: DPCT1003:0: Migrated API does not
return error code. (*, 0) is inserted. You may need to rewrite this code.
    status = cudaMemcpy(Result, d_C, VECTOR_SIZE*sizeof(float), cudaMemcpyDeviceToHost);
             ^
Processed 1 file(s) in -in-root folder "/home/eiforamr/data_samples"


See Diagnostics Reference to resolve warnings and complete the migration:
https://www.intel.com/content/www/us/en/develop/documentation/intel-dpcpp-compatibility-tool-
user-guide/top/diagnostics-reference.html


root@edgesoftware:/home/eiforamr/data_samples#
```

b. Conversion successfully done:

```
ls
dpct_output vector_add.cu
```

c. Go to output directory:

```
cd /dpct_output
```

d. Create a simple Makefile with this content:

```
CXX = dpcpp
TARGET = vector_add
SRCS = vector_add.dp.cpp


# Use predefined implicit rules and add one for *.cpp files.
%.o: %.cpp
	$(CXX) -c $(CXXFLAGS) $(CPPFLAGS) $< -o $@


all: $(TARGET)


$(TARGET): $(SRCS) $(DEPS)
	$(CXX) $(SRCS) -o $@


run: $(TARGET)
	./$(TARGET)
```

```
.PHONY: clean
clean:
	rm -f $(TARGET) *.o
```

e. Run make and then the output binary named `vector_add`:

```
make
./vector_add
```

Expected output:

A block of even numbers are listed, indicating the result of adding two vectors: [1..N] + [1..N].

```
./vector_add

2   4   6   8  10  12  14  16  18  20  22  24  26  28  30  32
34  36  38  40  42  44  46  48  50  52  54  56  58  60  62  64
66  68  70  72  74  76  78  80  82  84  86  88  90  92  94  96
98 100 102 104 106 108 110 112 114 116 118 120 122 124 126 128
130 132 134 136 138 140 142 144 146 148 150 152 154 156 158 160
162 164 166 168 170 172 174 176 178 180 182 184 186 188 190 192
194 196 198 200 202 204 206 208 210 212 214 216 218 220 222 224
226 228 230 232 234 236 238 240 242 244 246 248 250 252 254 256
258 260 262 264 266 268 270 272 274 276 278 280 282 284 286 288
290 292 294 296 298 300 302 304 306 308 310 312 314 316 318 320
322 324 326 328 330 332 334 336 338 340 342 344 346 348 350 352
354 356 358 360 362 364 366 368 370 372 374 376 378 380 382 384
386 388 390 392 394 396 398 400 402 404 406 408 410 412 414 416
418 420 422 424 426 428 430 432 434 436 438 440 442 444 446 448
450 452 454 456 458 460 462 464 466 468 470 472 474 476 478 480
482 484 486 488 490 492 494 496 498 500 502 504 506 508 510 512
```

Troubleshooting

The Makefile from step 9.d contains tabs and may not copy well to your system, giving this error:

```
Makefile:8: *** missing separator.  Stop.
```

To fix this, make sure there are tabs in lines 8, 15, 19, and 24.

For general robot issues, go to: Troubleshooting for Robot Tutorials.

Run Profiling Application in Docker* Container with VTune™ Profiler

Run the Sample Application

1. Go to the `AMR_containers` folder:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_containers
```

2. Prepare environment setup:

```
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
export CONTAINER_BASE_PATH=`pwd`
export ROS_DOMAIN_ID=19
```

3. Run the VTune™ profiler:

```
CHOOSE_USER=root docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/vtune.demo.yml
up oneapi
```

Expected output:

```
vtune: Warning: To profile kernel modules during the session, make sure they are available in
the /lib/modules/kernel_version/ location.
vtune: Collection started. To stop the collection, either press CTRL-C or enter from another
console window: vtune -r /tmp/matrix_multiply_vtune/r001gh -command stop.
Address of buf1 = 0x7f4578e4b010
Offset of buf1 = 0x7f4578e4b180
Address of buf2 = 0x7f457864a010
Offset of buf2 = 0x7f457864a1c0
Address of buf3 = 0x7f45746e2010
Offset of buf3 = 0x7f45746e2100
Address of buf4 = 0x7f4573ee1010
Offset of buf4 = 0x7f4573ee1140
Using multiply kernel: multiply1
Running on Intel(R) Iris(R) Xe Graphics [0x9a49]
Elapsed Time: 0.91916s
vtune: Collection stopped.
vtune: Using result path `/tmp/matrix_multiply_vtune/r001gh'
vtune: Executing actions 19 % Resolving information for `libpi_opencl.so'
vtune: Warning: Cannot locate debugging information for file `/usr/local/lib/
libze_intel_gpu.so.1'.
vtune: Executing actions 20 % Resolving information for `libc-dynamic.so'
vtune: Warning: Cannot locate debugging information for file `/lib/modules/5.10.65/kernel/fs/
overlayfs/overlay.ko'.
vtune: Executing actions 20 % Resolving information for `libm-2.31.so'
vtune: Warning: Cannot locate debugging information for file `/usr/lib/x86_64-linux-gnu/
libm-2.31.so'.
vtune: Executing actions 20 % Resolving information for `libc-2.31.so'
vtune: Warning: Cannot locate debugging information for file `/usr/lib/x86_64-linux-gnu/
libc-2.31.so'.
vtune: Executing actions 20 % Resolving information for `ld-2.31.so'
vtune: Warning: Cannot locate debugging information for file `/usr/lib/x86_64-linux-gnu/
ld-2.31.so'.
vtune: Warning: Cannot locate file `vmlinux'.
vtune: Executing actions 20 % Resolving information for `libpin3dwarf.so'
vtune: Warning: Cannot locate debugging information for file `/usr/local/lib/libigc.so.1.0.8517'.
vtune: Executing actions 20 % Resolving information for `libxed.so'
vtune: Warning: Cannot locate debugging information for the Linux kernel. Source-level analysis
will not be possible. Function-level analysis will be limited to kernel symbol tables. See the
Enabling Linux Kernel Analysis topic in the product online help for instructions.
vtune: Executing actions 21 % Resolving information for `libgcc_s.so.1'
vtune: Warning: Cannot locate debugging information for file `/usr/lib/x86_64-linux-gnu/
libgcc_s.so.1'.
vtune: Executing actions 21 % Resolving information for `libstdc++.so.6.0.28'
vtune: Warning: Cannot locate debugging information for file `/usr/lib/x86_64-linux-gnu/libstdc+
+.so.6.0.28'.
vtune: Executing actions 21 % Resolving information for `libtpsstool.so'
vtune: Warning: Cannot locate debugging information for file `/opt/intel/oneapi/vtune/2022.0.0/
lib64/libtpsstool.so'.
vtune: Executing actions 21 % Resolving information for `i915.ko'
vtune: Warning: Cannot locate debugging information for file `/opt/intel/oneapi/vtune/2022.0.0/
lib64/runtime/libittnotify_collector.so'.
vtune: Warning: Cannot locate debugging information for file `/opt/intel/oneapi/vtune/2022.0.0/
lib64/runtime/libittnotify_collector.so'.
vtune: Executing actions 22 % Resolving information for `libOpenCL.so.1'
vtune: Warning: Cannot locate debugging information for file `/usr/local/lib/
libze_intel_gpu.so.1.2.20939'.
vtune: Executing actions 22 % Resolving information for `libigdrcl.so'
```

```
vtune: Warning: Cannot locate debugging information for file `/lib/modules/5.10.65/kernel/
drivers/gpu/drm/i915/i915.ko'.
vtune: Warning: Cannot locate debugging information for file `/usr/local/lib/intel-opencl/
libigdrcl.so'.
vtune: Warning: Cannot locate debugging information for file `/usr/local/lib/intel-opencl/
libigdrcl.so'.
vtune: Executing actions 75 % Generating a report                              Elapsed Time:
1.163s
   GPU Time: 0.041s
EU Array Stalled/Idle: 55.0% of Elapsed time with GPU busy
| The percentage of time when the EUs were stalled or idle is high, which has a
| negative impact on compute-bound applications.
|
   GPU L3 Bandwidth Bound: 82.0% of peak value
   | L3 bandwidth was high when EUs were stalled or idle. Consider improving
   | cache reuse.
   |

      Hottest GPU Computing Tasks Bound by GPU L3 Bandwidth
      Computing Task  Total Time
      --------------  ----------
      Matrix1<float>      0.035s
   Occupancy: 91.1% of peak value

      Hottest GPU Computing Tasks with Low Occupancy
      Computing Task  Total Time  SIMD Width  Peak Occupancy(%)  Occupancy(%)  SIMD
Utilization(%)
      --------------  ----------  ----------  -----------------  ------------
-------------------
   Sampler Busy: 0.0% of peak value

      Hottest GPU Computing Tasks with High Sampler Usage
      Computing Task  Total Time
      --------------  ----------
Collection and Platform Info
   Application Command Line: ./matrix.dpcpp
   Operating System: 5.10.65 DISTRIB_ID=Ubuntu DISTRIB_RELEASE=20.04 DISTRIB_CODENAME=focal
DISTRIB_DESCRIPTION="Ubuntu 20.04.3 LTS"
   Computer Name: glaic3aeon2
   Result Size: 28.3 MB
   Collection start time: 15:39:14 04/01/2022 UTC
   Collection stop time: 15:39:15 04/01/2022 UTC
   Collector Type: Event-based sampling driver,Driverless Perf system-wide sampling,User-mode
sampling and tracing
   CPU
      Name: Intel(R) microarchitecture code named Tigerlake
      Frequency: 2.803 GHz
      Logical CPU Count: 8
   GPU
      Name: TigerLake GT2 [Iris Xe Graphics]
      Vendor: Intel Corporation
      EU Count: 96
      Max EU Thread Count: 7
      Max Core Frequency: 1.350 GHz
      GPU OpenCL Info
            Version
            Max Compute Units: 96
            Max Work Group Size: 512
            Local Memory: 65.5 KB
```

```
            SVM Capabilities

If you want to skip descriptions of detected performance issues in the report,
enter: vtune -report summary -report-knob show-issues=false -r <my_result_dir>.
Alternatively, you may view the report in the csv format: vtune -report
<report_name> -format=csv.
vtune: Executing actions 100 % done
```

4. For a list of the steps that were executed, see `01_docker_sdk_env/docker_compose/05_tutorials/vtune.demo.yml`.

Troubleshooting

For general robot issues, go to: Troubleshooting for Robot Tutorials.

Run OpenVINO™ Benchmarking Tool

This tutorial tells you how to run the benchmark application on an 11th Generation Intel® Core™ processor with an integrated GPU. It uses the asynchronous mode to estimate deep learning inference engine performance and latency.

Start Docker* Container

1. Go to the AMR_containers folder:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_containers
```

2. Start the Docker container as root:

```
./run_interactive_docker.sh eiforamr-full-flavour-sdk:<TAG> root
```

Set Environment Variables

The environment variables must be set before you can compile and run OpenVINO™ applications.

1. Run the following script:

```
source /opt/intel/openvino/bin/setupvars.sh
             --or--
source <OPENVINO_INSTALL_DIR>/bin/setupvars.sh
```

Build Benchmark Application

1. Change directory and build the benchmark application using the `cmake` script file using the following commands:

```
cd /opt/intel/openvino/inference_engine/samples/cpp
./build_samples.sh
```

2. Once the build is successful, access the benchmark application in the following directory:

```
cd /root/inference_engine_cpp_samples_build/intel64/Release
       -- or --
cd <INSTALL_DIR>/inference_engine_cpp_samples_build/intel64/Release
```

   The `benchmark_app` application is available inside the Release folder.

Input File

Select an image file or a sample video file to provide an input to the benchmark application from the following directory:

```
cd /root/inference_engine_cpp_samples_build/intel64/Release
```

Application Syntax and Options

The benchmark application syntax is as follows:

```
./benchmark_app [OPTION]
```

In this tutorial, we recommend you select the following options:

```
./benchmark_app -m <model> -i <input> -d <device> -nireq <num_reqs> -nthreads <num_threads> -b
<batch>


where:
<model>-------------The complete path to the model .xml file
<input>-------------The path to the folder containing image or sample video file.
<device>------------The device type can be GPU or CPU etc.,
<num_reqs>----------No of parallel inference requests
<num_threads>-------No of threads to use for inference on the CPU (throughput mode)
<batch>-------------Batch size
```

For complete details on the available options, run the following command:

```
./benchmark_app -h
```

Run the Application

The benchmark application is executed as seen below. This tutorial uses the following settings:

- Benchmark application is executed on `frozen_inference_graph` model.
- Number of parallel inference requests is set as 8.
- Number of CPU threads to use for inference is set as 8.
- Device type is GPU.

```
./benchmark_app -d GPU -i ~/<dir>/input/ -m /home/eiforamr/workspace/object_detection/src/
object_detection/models/ssd_mobilenet_v2_coco/frozen_inference_graph.xml -nireq 8 -nthreads 8
./benchmark_app -d GPU -i /home/eiforamr/data_samples/media_samples/plates_720.mp4 -m /home/
eiforamr/workspace/object_detection/src/object_detection/models/ssd_mobilenet_v2_coco/
frozen_inference_graph.xml -nireq 8 -nthreads 8
```

Expected output:

```
[Step 1/11] Parsing and validating input arguments
[ INFO ] Parsing input parameters
[ INFO ] Files were added: 1
[ INFO ]     /home/eiforamr/data_samples/media_samples/plates_720.mp4
[Step 2/11] Loading Inference Engine
[ INFO ] InferenceEngine:
        API version ............ 2.1
        Build .................. 2021.2.0-1877-176bdf51370-releases/2021/2
        Description ....... API
[ INFO ] Device info:
        GPU
        clDNNPlugin version ......... 2.1
        Build ........... 2021.2.0-1877-176bdf51370-releases/2021/2


[Step 3/11] Setting device configuration
[ WARNING ] -nstreams default value is determined automatically for GPU device. Although the
automatic selection usually provides a reasonable performance,but it still may be non-optimal
for some cases, for more information look at README.
[Step 4/11] Reading network files
[ INFO ] Loading network files
```

```
[ INFO ] Read network took 89.49 ms
[Step 5/11] Resizing network to match image sizes and given batch
[ INFO ] Network batch size: 1
[Step 6/11] Configuring input of the model
[Step 7/11] Loading the model to the device
[ INFO ] Load network took 44714.68 ms
[Step 8/11] Setting optimal runtime parameters
[Step 9/11] Creating infer requests and filling input blobs with images
[ INFO ] Network input 'image_tensor' precision U8, dimensions (NCHW): 1 3 300 300
[ WARNING ] No supported image inputs found! Please check your file extensions: bmp, dib, jpeg,
jpg, jpe, jp2, png, pbm, pgm, ppm, sr, ras, tiff, tif
[ INFO ] Infer Request 0 filling
[ INFO ] Fill input 'image_tensor' with random values (image is expected)
[ INFO ] Infer Request 1 filling
[ INFO ] Fill input 'image_tensor' with random values (image is expected)
[ INFO ] Infer Request 2 filling
[ INFO ] Fill input 'image_tensor' with random values (image is expected)
[ INFO ] Infer Request 3 filling
[ INFO ] Fill input 'image_tensor' with random values (image is expected)
[ INFO ] Infer Request 4 filling
[ INFO ] Fill input 'image_tensor' with random values (image is expected)
[ INFO ] Infer Request 5 filling
[ INFO ] Fill input 'image_tensor' with random values (image is expected)
[ INFO ] Infer Request 6 filling
[ INFO ] Fill input 'image_tensor' with random values (image is expected)
[ INFO ] Infer Request 7 filling
[ INFO ] Fill input 'image_tensor' with random values (image is expected)
[Step 10/11] Measuring performance (Start inference asynchronously, 8 inference requests using 2
streams for GPU, limits: 60000 ms duration)
[ INFO ] First inference took 10.01 ms


[Step 11/11] Dumping statistics report
Count:      9456 iterations
Duration:   60066.11 ms
Latency:    51.33 ms
Throughput: 157.43 FPS
```

Benchmark Report

Sample execution results using an 11th Gen Intel® Core™ i7-1185GRE @ 2.80 GHz.

Read network time (ms)	89
Load network time (ms)	44714.68
First inference time (ms)	10.01
Total execution time (ms)	60066.11
Total num of iterations	9456
Latency (ms)	51.33
Throughput (FPS)	157.43

NOTE Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. No product or component can be absolutely secure. Performance varies by use, configuration and other factors. Learn more at Intel® Performance Index.

Troubleshooting

For general robot issues, go to: Troubleshooting for Robot Tutorials.

Run the ADBSCAN Algorithm

This tutorial tells you how to run the ADBSCAN algorithm from EI for AMR using 2D Slamtec* RPLIDAR and Intel® RealSense™ camera input.

It outputs to the `obstacle_array` topic of type `nav2_dynamic_msgs/ObstacleArray`.

Prerequisites: You know how to connect and configure a Slamtec* RPLIDAR sensor. For details, see: Launch Cartographer with 2D LIDAR.

Run the ADBSCAN Algorithm with Slamtec* RPLIDAR Input

1. Go to the `AMR_containers` folder:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_*/AMR_containers
```

2. Prepare the environment setup:

```
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
export CONTAINER_BASE_PATH=`pwd`
export ROS_DOMAIN_ID=17
# Unzip the ros2 bags if they were not unzipped before
unzip 01_docker_sdk_env/docker_compose/06_bags.zip  -d 01_docker_sdk_env/docker_compose/
```

3. Depending on the Slamtec* RPLIDAR availability, you have two possibilities:

   - Slamtec* RPLIDAR connected

     Start a pre-configured yml file that starts the LIDAR Node and then the ADBSCAN application:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
adbscan_LIDAR.tutorial.yml up
```

   - No Slamtec* RPLIDAR connected

     Start a pre-configured yml file that plays a ROS 2 bag containing LIDAR data and then the ADBSCAN application:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
adbscan_2D.tutorial.yml up
```

     Expected output: ADBSCAN prints logs of its interpretation of the LIDAR data coming from the ROS 2 bag.

```
amr-adbscan | [INFO] [1656585216.649840684] [adbscan_sub_node]: valid data p
amr-adbscan | num_points: 1232
amr-adbscan | 0. test CPU: ADBScan execution time: 0.0156487 [s]
amr-adbscan | number of clusters:22
amr-adbscan | [INFO] [1656585217.227701651] [adbscan_sub_node]: Msg: number
amr-adbscan | Lidar Message Received
amr-adbscan | [INFO] [1656585217.227984055] [adbscan_sub_node]: valid data p
amr-adbscan | num_points: 1285
amr-adbscan | 0. test CPU: ADBScan execution time: 0.0174741 [s]
amr-adbscan | number of clusters:22
amr-adbscan | Lidar Message Received
amr-adbscan | [INFO] [1656585217.772819965] [adbscan_sub_node]: Msg: number
amr-adbscan | [INFO] [1656585217.773052061] [adbscan_sub_node]: valid data p
amr-adbscan | num_points: 1289
amr-adbscan | 0. test CPU: ADBScan execution time: 0.0165351 [s]
amr-adbscan | number of clusters:17
amr-adbscan | Lidar Message Received
amr-adbscan | [INFO] [1656585218.283061466] [adbscan_sub_node]: Msg: number
amr-adbscan | [INFO] [1656585218.283283853] [adbscan_sub_node]: valid data p
amr-adbscan | num_points: 1310
amr-adbscan | 0. test CPU: ADBScan execution time: 0.0171757 [s]
amr-adbscan | number of clusters:20
amr-adbscan | [INFO] [1656585218.827898998] [adbscan_sub_node]: Msg: number
amr-adbscan | Lidar Message Received
amr-adbscan | [INFO] [1656585218.828191588] [adbscan_sub_node]: valid data p
amr-adbscan | num_points: 1288
amr-adbscan | 0. test CPU: ADBScan execution time: 0.0182069 [s]
amr-adbscan | number of clusters:20
```

Run the ADBSCAN Algorithm with Intel® RealSense™ Camera Input

1. Go to the `AMR_containers` folder:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_containers
```

2. Prepare the environment setup:

```
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
export CONTAINER_BASE_PATH=`pwd`
export ROS_DOMAIN_ID=17
# Unzip the ros2 bags if they were not unzipped before
unzip 01_docker_sdk_env/docker_compose/06_bags.zip  -d 01_docker_sdk_env/docker_compose/
```

3. Depending on the Intel® RealSense™ camera availability, you have two possibilities:

   - Intel® RealSense™ camera connected

     Start a pre-configured yml file that starts the Intel® RealSense™ node and then the ADBSCAN application:

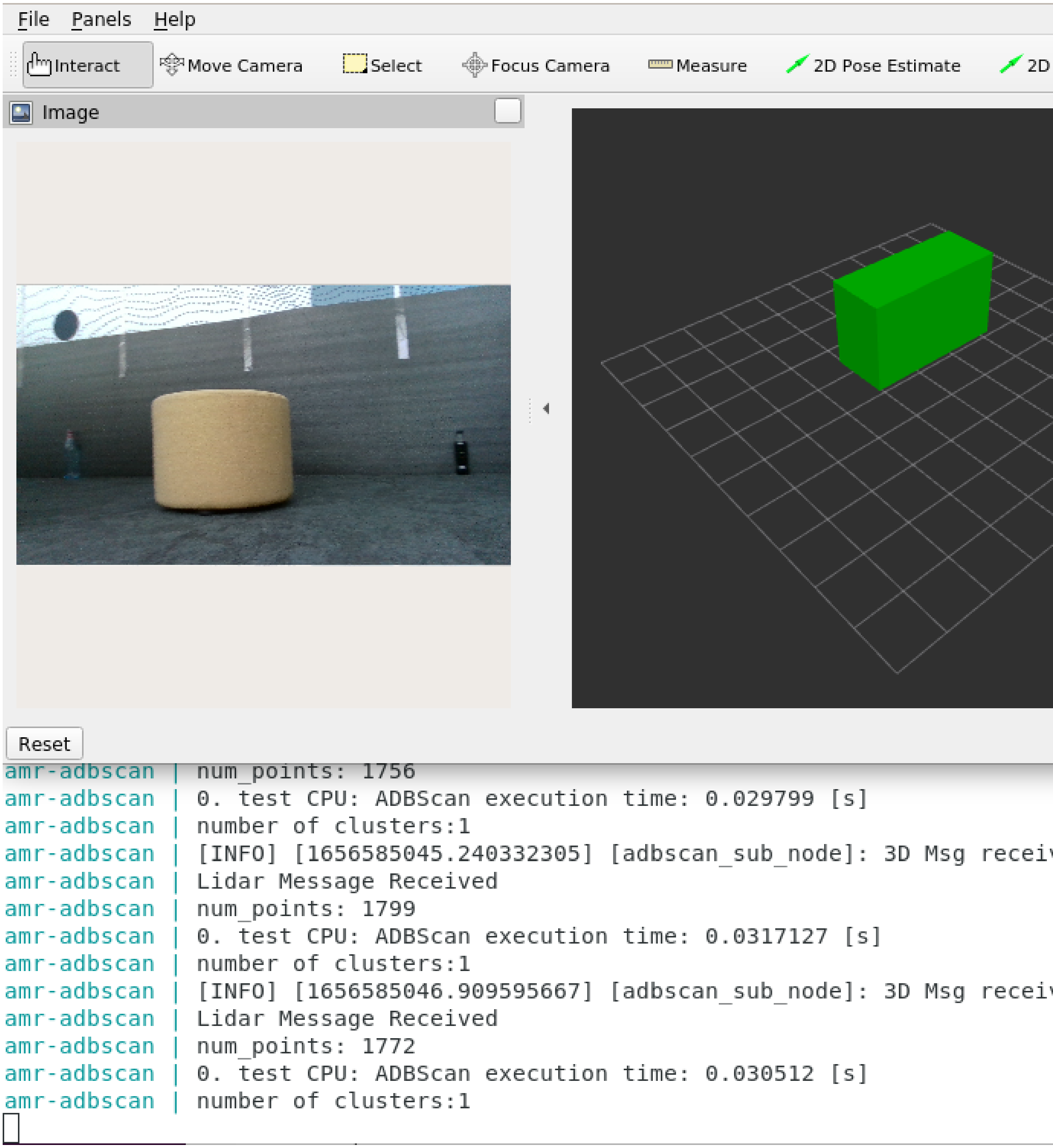
```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
adbscan_RealSense.tutorial.yml up
```

   - No Intel® RealSense™ camera connected

Start a pre-configured yml file that plays a ROS 2 bag containing Intel® RealSense™ data and then the ADBSCAN application:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
adbscan_RS.tutorial.yml up
```

Expected result: rviz2 starts, and you see how ADBSCAN interprets Intel® RealSense™ data coming from the ros2 bag:



Troubleshooting

For general robot issues, go to: Troubleshooting for Robot Tutorials.

Launch Wandering Application with Gazebo* Simulation

This tutorial tells you how to use an industrial simulation room model (the OSRF GEAR workcell that was used for the 2018 ARIAC competition) with objects and a wafflebot3 robot for simulation in Gazebo*. The industrial room includes: shelves, conveyor belts, pallets, boxes, robots, stairs, ground lane markers, and a tiled boundary wall.

Run the Sample Application

1. If your system has an Intel® GPU, follow the steps in the Get Started Guide for Robots to enable the GPU for simulation. This step improves Gazebo* simulation performance.
2. Go to the `AMR_containers` folder:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_containers
```

3. Prepare environment setup:

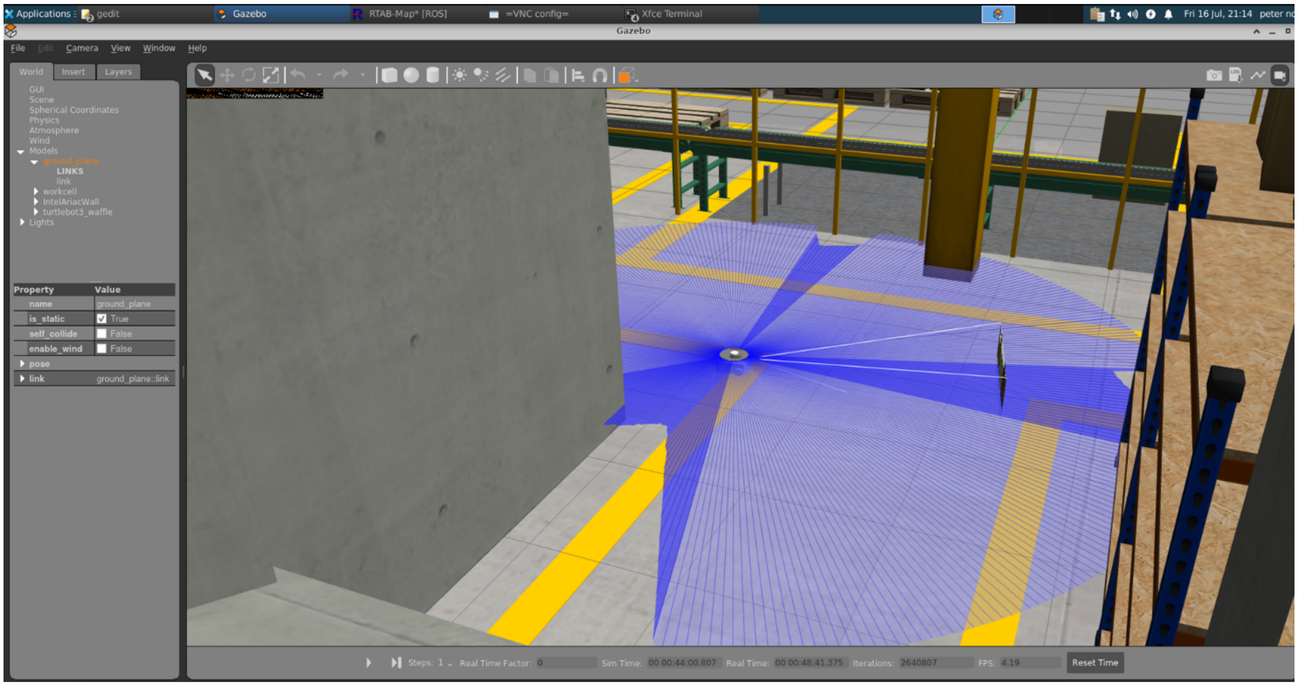
```
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
export CONTAINER_BASE_PATH=`pwd`
export ROS_DOMAIN_ID=32
```

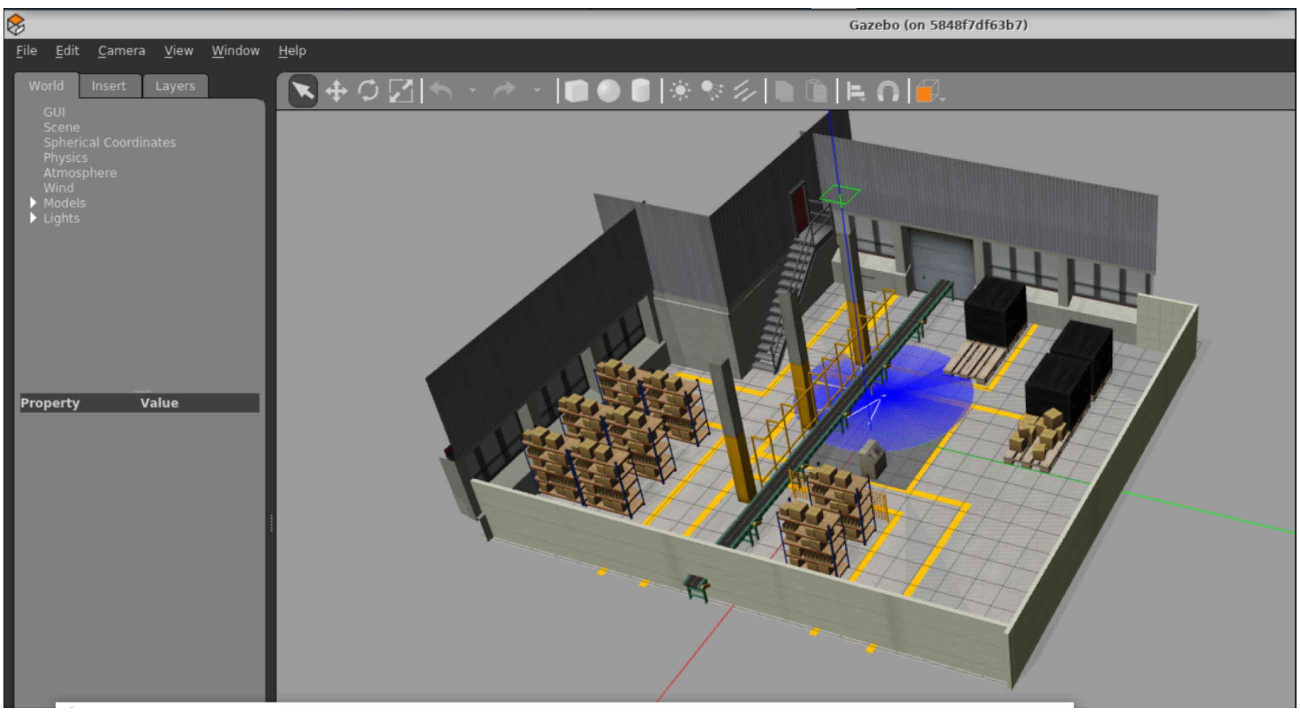
4. Run the command below to start the Docker container:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
wandering_gazebo_ariac.demo.yml up
```

   Expected output:

   The robot starts wandering inside the simulation. See the simulation snapshots from different angles:





To increase performance, the real time update rate can be set to 0:

- On Gazebo, in the left panel, go to the **World** Tab and click on **Physics**.
- Change the real time update rate to 0.

5. To close this, do one of the following:

- Type `Ctrl-c` in the terminal where you did the up command.
- Run this command in another terminal:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
wandering_gazebo_ariac.demo.yml down
```

Troubleshooting

If the robot is not moving but Gazebo* is started, start the Wandering application manually by opening a container shell and entering:

```
docker-compose -f 01_docker_sdk_env/docker_compose/01_amr/amr-sdk.all.yml run wandering bash
ros2 run wandering_app wandering
```

For general robot issues, go to: Troubleshooting for Robot Tutorials.

Launch Wandering Application on AAEON* Robot Kit

This tutorial tells you how to:

- Assemble and calibrate your robot.
- Map an area using the wandering application and Aaeon's UP Xtreme i11 Robotic Kit.
- Test your AAEON* robot kit using ros2_amr_interface and teleop_twist_keyboard.

Hardware Prerequisite

You need one of the following AAEON* robot kits:

- UPS 6000 Robotic Kit
- UP Xtreme i11 Robotic Kit

This tutorial uses the UP Xtreme i11 Robotic Kit.

Additional AAEON* resources:

- Development Kit: https://github.com/up-board/up-community/wiki/UP-Robotic-Development-Kit-QSG
- Hardware Assembly: https://github.com/up-board/up-community/wiki/UP-Robotic-Development-Kit-HW-Assembly-Guide
- Power Management: https://github.com/up-board/up-community/wiki/UP-Robotic-Development-Kit-Power-Management-Guide

Calibrate your Robot's Inertial Measurement Unit (IMU) Sensor

The IMU sensor is used to determine the robot's orientation. Moving the robot interferes with calibration, so do not move the robot while performing these steps.

1. Prepare the environment:

```
mkdir ~/imu_cal
sudo chmod 0777 ~/imu_cal
cd <path to where the EI for AMR was installed>Edge_Insights_for_Autonomous_Mobile_Robots_*/
AMR_containers/
```

2. Start the ros2_amr_interface:

```
./run_interactive_docker.sh amr-aaeon-amr-interface:2022.2 eiforamr -c imu_cal
export ROS_DOMAIN_ID=27
ros2 run ros2_amr_interface amr_interface_node --ros-args -p try_reconnect:=true -p
publishTF:=true --remap /amr/cmd_vel:=/cmd_vel -p port_name:=/dev/ttyUSB0
```

The output should be similar to:

```
[INFO] [1655311130.138413471] [IoContext::IoContext]: Thread(s) Created: 2
[INFO] [1655311130.139098979] [AMR_node]: Serial opened on /dev/ttyUSB0 at 115200
[INFO] [1655311131.144572706] [AMR_node]: Hardware is now online
```

If it is not, verify that the motor Controller is not on /dev/ttyUSB0. Adapt the command accordingly.

3. In another terminal, attach to the opened Docker* image, and get the data needed for calibration.

```
docker exec -it imu_cal bash
source ros_entrypoint.sh
export ROS_DOMAIN_ID=27
ros2 topic echo /amr/imu/raw | grep velocity: -A 3 | grep '[xyz]:' | grep -v velocity >
```

```
velocity.txt
# Wait for at least 5 seconds and then press ctrl-c
cp velocity.txt /home/<user>/imu_cal  # replace <user> with your host's user.
exit
```

4. Open the first terminal, type `Ctrl-c` to stop amr_interface_node, and exit the Docker* image.

```
ctrl-c
exit
```

5. Go to the imu_cal folder, and create the get_offset.sh script:

```
cd ~/imu_cal
gedit get_offset.sh # you can use vim/nano or any text editor is preferred.
# copy-paste the following snippet of code in this script:
##########
!/bin/bash

filename=$1
i=0
res=0

if [ "$2" != "x" ] && [ "$2" != "y" ] && [ "$2" != "z" ] || [ "$#" -ne 2 ]
then
   echo "Usage:      $0 imu_samples_file axis"
   echo "Example:    $0 samples.txt z"
   exit 1
fi

len=`cat $filename | wc -l`
if [ $len -lt 450 ]
then
   echo "Error: Previous command was interrupted too soon. Please let it run for at least five 
seconds."
   exit 2
fi

while read line
do
   isdata=`echo $line | grep "$2" | wc -l`
   if [ $isdata -gt 0 ]
   then
                i=$(($i+1))
                if [ $i -gt 100 ]
                then
                            i=100
                            break;
                else
                   line=`echo $line | awk '{print $2}'`
                      res=$( bc <<< "$res - $line")
                fi
   fi
done < $filename

res=$(echo "$res/$i"|bc -l)

echo "Offset for $2 axis is $res"

##########
```

6. Get the calibration values for the X, Y and Z axes:

```
chmod a+x ./get_offset.sh
./get_offset.sh velocity.txt x
./get_offset.sh velocity.txt y
./get_offset.sh velocity.txt z
```

Example of output:

```
./get_offset.sh velocity.txt x
Offset for x axis is -.00280559304170310501
./get_offset.sh velocity.txt y
Offset for y axis is -.00537949499936075887
./get_offset.sh velocity.txt z
Offset for z axis is .00155259681894676760
```

7. Put these values in aaeon_node_params.yaml:

```
cd <path to where the EI for AMR was installed>Edge_Insights_for_Autonomous_Mobile_Robots_*/
AMR_containers/
gedit 01_docker_sdk_env/docker_compose/05_tutorials/param/aaeon_node_params.yaml  # you can use
vim/nano or any text editor is preferred.
# Use the values you got for the x, y and z axes to offsets for gyro. Using the values from
above our example would look like so:
    imu:
      frame_id: imu_link
      offsets:
      #     accelerometer:
      #       x: 0.0
      #       y: 0.0
      #       z: 0.0
         gyro:
           x: -0.00280559304170310501
           y: -0.00537949499936075887
           z: 0.00155259681894676760
```

NOTE Indentation is important in yaml files, so make sure to align `offsets` with `frame_id`. If the indentation is incorrect, the container reports an error when started.

Map an Area using the Wandering Application and the UP Xtreme i11 Robotic Kit

The goal of the wandering application is to map an area and avoid hitting objects.

1. Place the robot in an area with multiple objects in it.

2. Go to the installation folder of Edge_Insights_for_Autonomous_Mobile_Robots:

```
cd <path to where the EI for AMR was installed>Edge_Insights_for_Autonomous_Mobile_Robots_*/
AMR_containers/
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
export CONTAINER_BASE_PATH=`pwd`
export ROS_DOMAIN_ID=27
```

3. Start mapping the area:

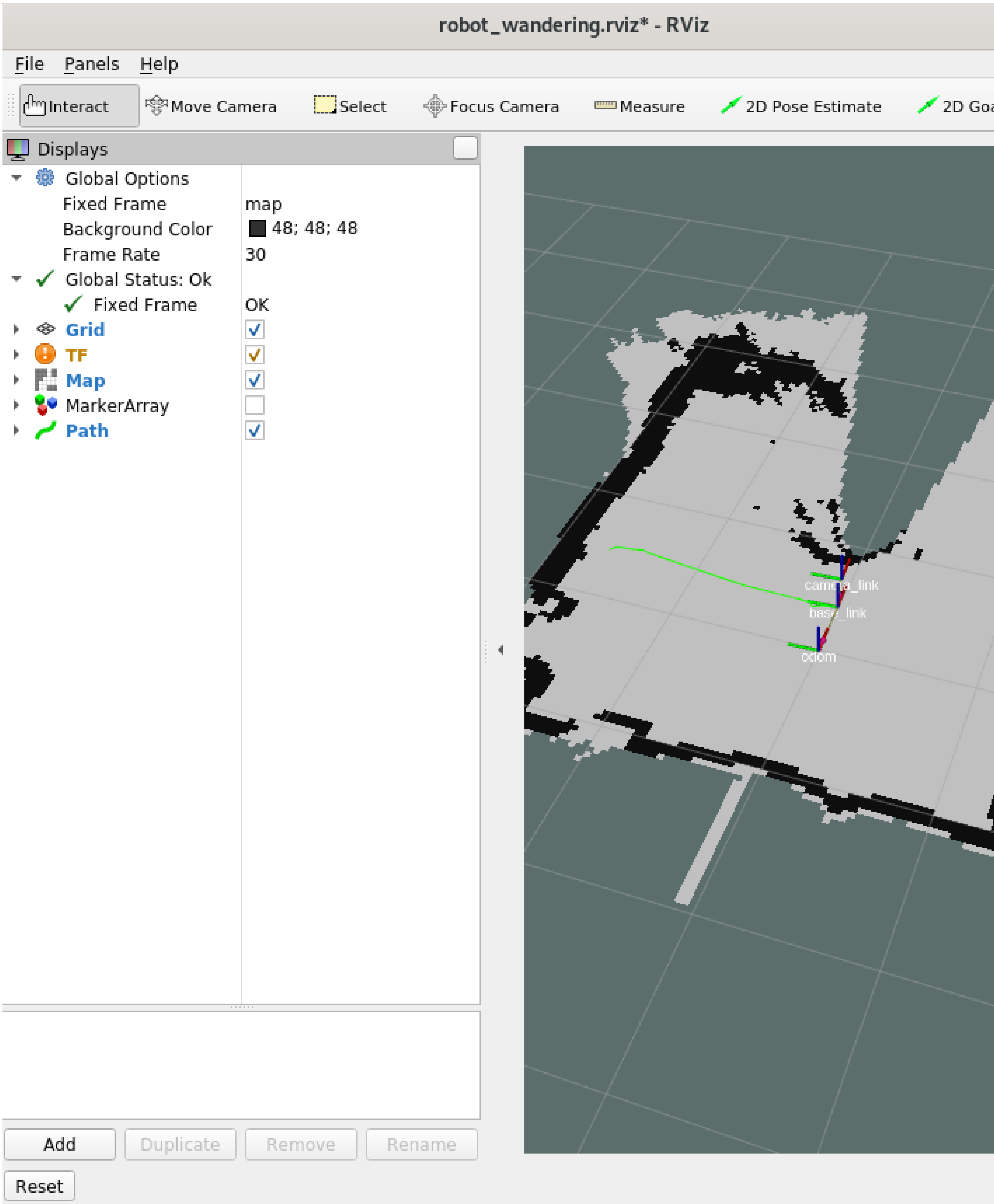
```
docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
aaeon_wandering__aaeon_realsense_collab_slam_fm_nav2_ukf.tutorial.yml up
```

Expected result: The robot starts wandering around the room and mapping the entire area.

4. On a different terminal, prepare the environment to visualize the mapping and the robot using rviz2.

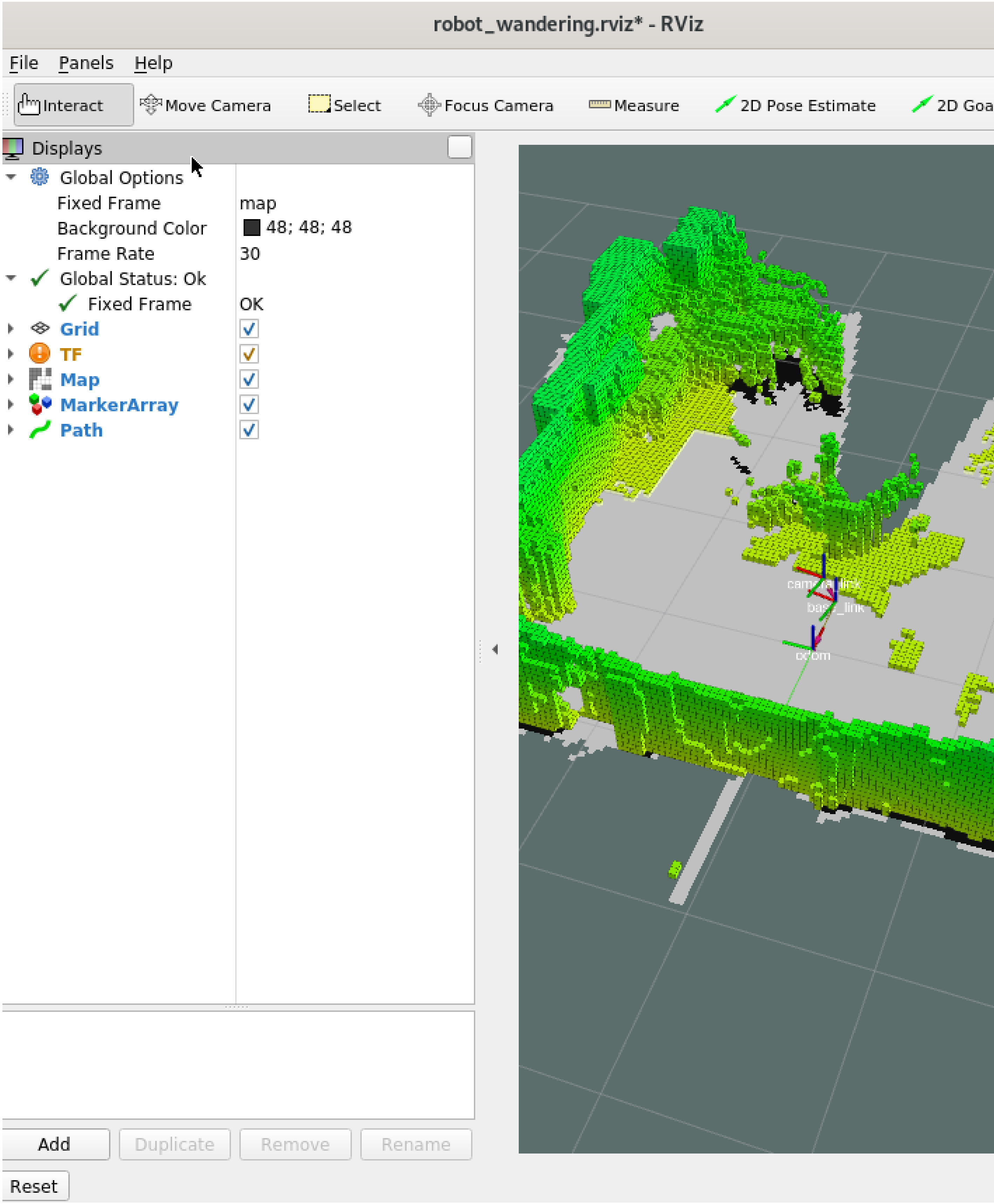
NOTE If available, use a different development machine because rviz2 consumes a lot of resources that may interfere with the robot.

```
cd <path to where the EI for AMR was installed>Edge_Insights_for_Autonomous_Mobile_Robots_*/
AMR_containers/
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
export CONTAINER_BASE_PATH=`pwd`
export ROS_DOMAIN_ID=27
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
rviz_robot_wandering.yml up
```



5. To see the map in 3D, can check the MarkerArray:

NOTE Displaying in 3D consumes a lot of the system resources. Intel recommends opening rviz2 on a development system. The development system needs to be in the same network and have the same ROS_DOMAIN_ID set.



6. To stop the robot from mapping the area, do one of the following:

- Type `Ctrl-c` in the terminal where the aaeon_wandering__aaeon_realsense_collab_slam_fm_nav2_ukf.tutorial.yml was run.
- (Preferred method because this option cleans the workspace) On the system you used docker-compose up on in step 2, use docker-compose down:

```
docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
aaeon_wandering__aaeon_realsense_collab_slam_fm_nav2_ukf.tutorial.yml down --remove-orphans
```

If the robot moves in an unpredictable way and hits objects easily, there may be some hardware configuration issues. See the Troubleshooting section for suggestions.

Control the UP Xtreme i11 Robotic Kit Using a Gamepad

This example uses a Logitech* F710 wireless gamepad.

1. Insert the USB dongle in the robot.
2. Go to the installation folder of Edge_Insights_for_Autonomous_Mobile_Robots, and prepare the environment:

```
cd <path to where the EI for AMR was installed>Edge_Insights_for_Autonomous_Mobile_Robots_*/
AMR_containers/
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
export CONTAINER_BASE_PATH=`pwd`
export ROS_DOMAIN_ID=27
sudo chmod a+rw /dev/input/js0
sudo chmod a+rw /dev/input/event*
```

3. Start mapping the area:

```
docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
aaeon_wandering__aaeon_realsense_collab_slam_fm_nav2_ukf.tutorial.yml up
```

4. After the robot starts to move, you can control it with the gamepad:
   - To control the robot, keep the top right button labeled 1 in the picture below pressed at all times.
   - To move the robot on the X and Y axes, enable the mode button, and use the buttons labeled 2 in the picture below.
   - To rotate the robot in place, use the joystick labeled 3 in the picture below.
   - To move the robot on the X and Y axrs, disable the mode button, and use the joystick labeled 4 in the picture below.

Control the UP Xtreme i11 Robotic Kit Using teleop_twist_keyboard

Test the AAEON* robot development kit to validate that the hardware setup was done correctly.

1. Go to the installation folder of Edge_Insights_for_Autonomous_Mobile_Robots, and prepare the environment:

```
cd <path to where the EI for AMR was installed>Edge_Insights_for_Autonomous_Mobile_Robots_*/
AMR_containers/
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
export CONTAINER_BASE_PATH=`pwd`
```

2. Open the aaeon-amr-interface image:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/01_amr/amr-sdk.all.yml
run aaeon-amr-interface bash
```

3. Open ros2_amr_interface:

```
source /home/eiforamr/workspace/ros_entrypoint.sh
export ROS_DOMAIN_ID=167
ros2 run ros2_amr_interface amr_interface_node --ros-args -p try_reconnect:=true -p
timeout_connection:=1000.0 -p publishTF:=true --remap /amr/cmd_vel:=/cmd_vel -p port_name:=/dev/
ttyUSB0
```

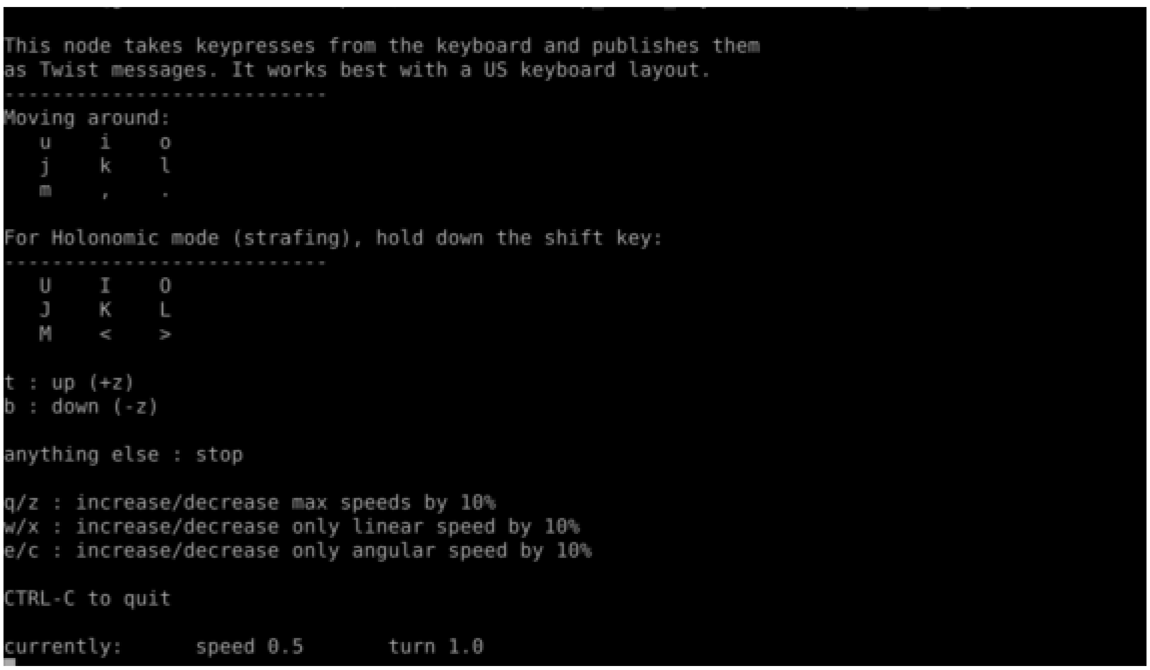
NOTE The above command uses `port_name:=/dev/ttyUSB0`. You may have a different value in your setup. To check the value in your setup, enter the command:

```
dmesg
```

In the output, search for text similar to this string: `usb 3-3: ch341-uart converter now attached to ttyUSB0` and update the command to match your setup.

4. In another terminal, open full-sdk, and start teleop_twist_keyboard:

```
cd <path to where the EI for AMR was installed>Edge_Insights_for_Autonomous_Mobile_Robots_*/
AMR_containers/
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
export CONTAINER_BASE_PATH=`pwd`
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/01_amr/amr-sdk.all.yml
run full-sdk bash
source /home/eiforamr/workspace/ros_entrypoint.sh
export ROS_DOMAIN_ID=167
ros2 run teleop_twist_keyboard teleop_twist_keyboard
```



Expected result: You can now control your robot to see if the hardware configuration is correct.

The AAEON* robotic development kit instructions explain that the robot should respond to your commands in these ways:

- Move forward when you press **i**
- Stop when you press **k**
- Move backward when you press **,**
- Stop when you press **k**
- Turn right when you press **j**
- Stop when you press **k**
- Strafe (move sideways) when you press **L** or **J**

Test your robot to confirm that it responds to your commands. If not, see the Troubleshooting section.

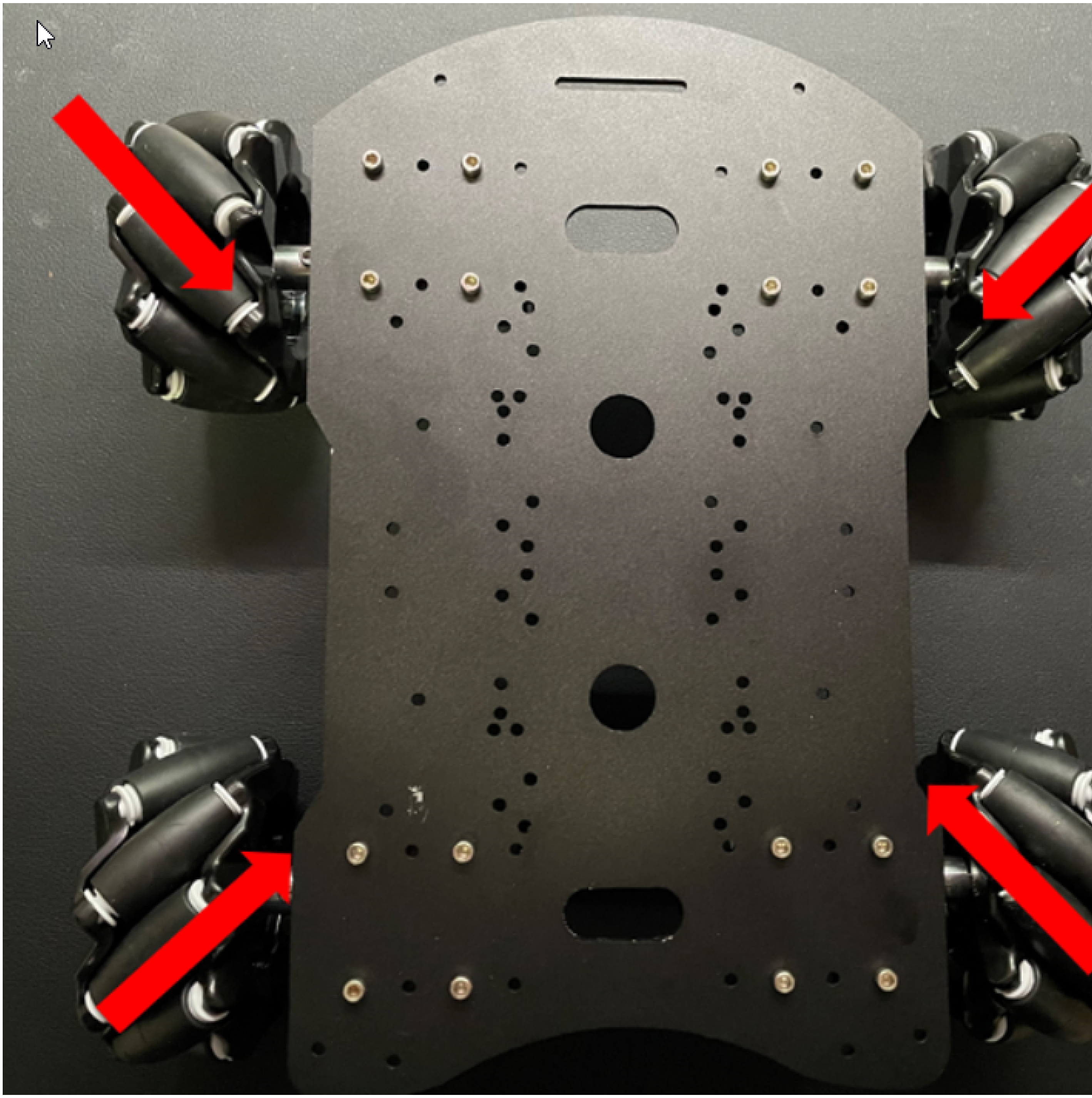
Troubleshooting

If the robot does not start moving, the firmware might be stuck. To make it work again:

```
cd <path to where the EI for AMR was installed>Edge_Insights_for_Autonomous_Mobile_Robots_*/
AMR_containers/
./run_interactive_docker.sh amr-aaeon-amr-interface:2022.2 eiforamr -c aaeon_robot
ros2 run ros2_amr_interface amr_interface_node --ros-args -p try_reconnect:=true -p
publishTF:=true --remap /amr/cmd_vel:=/cmd_vel -p port_name:=/dev/ttyUSB0
ctrl-c
ros2 run ros2_amr_interface amr_interface_node --ros-args -p try_reconnect:=true -p
publishTF:=true --remap /amr/cmd_vel:=/cmd_vel -p port_name:=/dev/ttyUSB0
# Look for the text: [INFO] [1655311131.144572706] [AMR_node]: Hardware is now online
# If you don't get this repeat the commands from the docker image and check if the motor
controller is not attached to /dev/ttyUSB0.
# If it is not attached to /dev/ttyUSB0, find out which one it is and adapt the commands
accordingly.
# When you get the [INFO] [1655311131.144572706] [AMR_node]: Hardware is now online, exit the
docker image:
exit
```

If the robot is not behaving as instructed when using the teleop_twist_keyboard, try the following steps.

1. Check the direction of the wheels. The way they are facing is very important, as shown in the picture below.

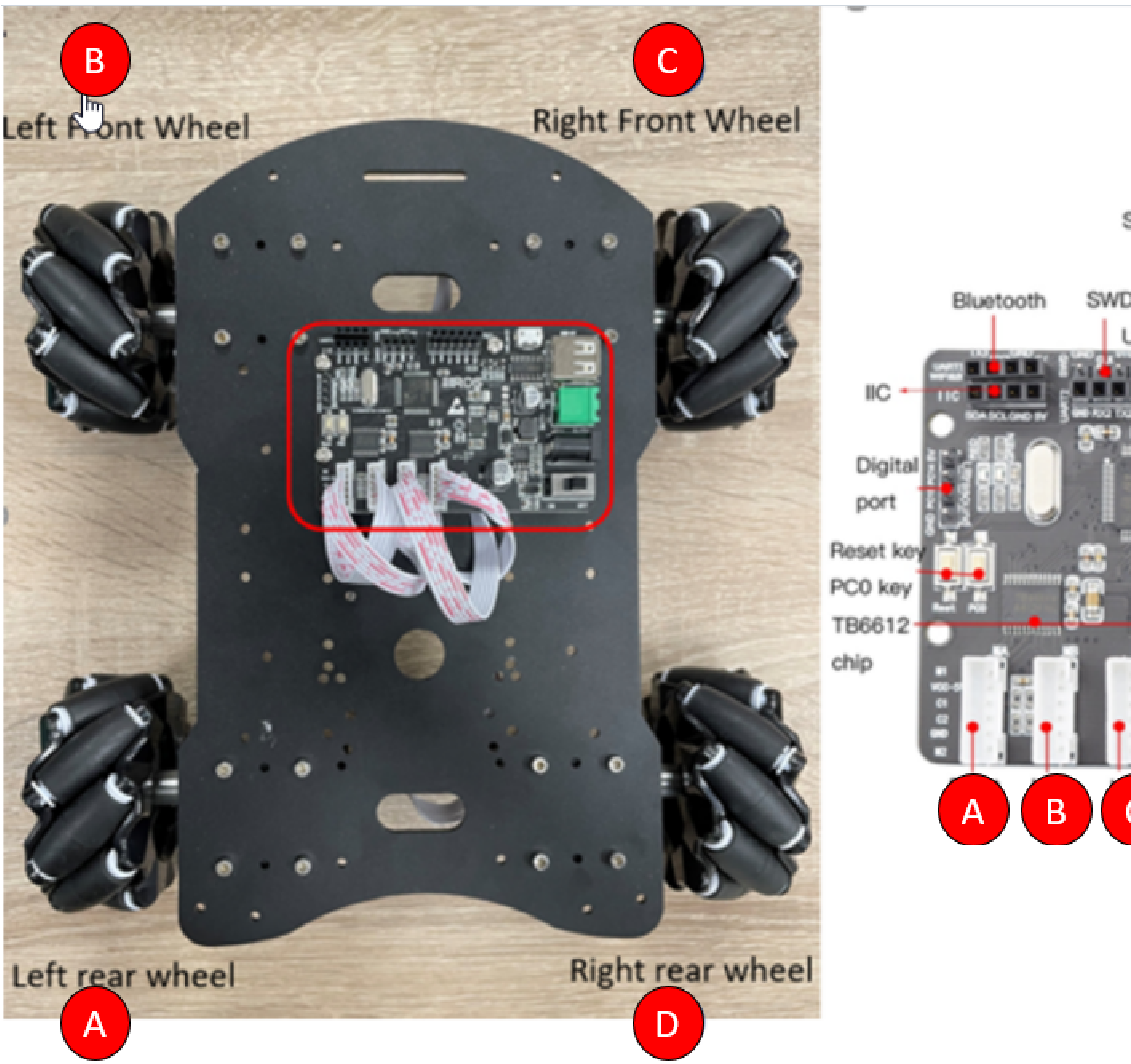


In our local setup we had to do the following Wheel setup, on each wheel you get a R (Right) or L (Left):

R(wheel) <<<>>> L(wheel)

L(wheel) <<<>>> R(wheel)

2. Check the connection between the wheels (left in the picture below) and the motor controller.



It is very important to have the hardware setup correctly configured. If it is not correct, it will be evident when testing with the teleop_twist_keyboard.

3. If the wheels do not turn at all, there may be something wrong with the wheel motor control. The board's datasheet states that it takes a 12 V input. Intel found that a 12.5 V input did not work, but 5V, 8V, and 10V inputs do work.

For general robot issues, go to: Troubleshooting for Robot Tutorials.

Launch Cartographer with 2D LIDAR

This tutorial demonstrates how Cartographer can work with different types of 2D LIDAR including:

- Slamtec* RPLIDAR A3 2D LIDAR
- Sick NanoScan3 Safety Lidar

Slamtec* RPLIDAR A3 2D LIDAR

1. Connect a Slamtec* RPLIDAR A3 2D LIDAR to your system.
2. Add a new udev rule.

   a. Create a new file:

```
sudo nano /etc/udev/rules.d/rplidar.rules
```

   b. Add these lines:

```
# set the udev rule, make the device_port be fixed by rplidar
#
KERNEL=="ttyUSB*", ATTRS{idVendor}=="10c4", ATTRS{idProduct}=="ea60", MODE:="0777", SYMLINK
+="rplidar"
```

   c. Reload the rules:

```
sudo udevadm control --reload-rules
sudo udevadm trigger
```

3. Run the Sample Application.

   a. Go to the `AMR_containers` folder:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_containers
```

   b. Prepare the `docker_compose` environment:

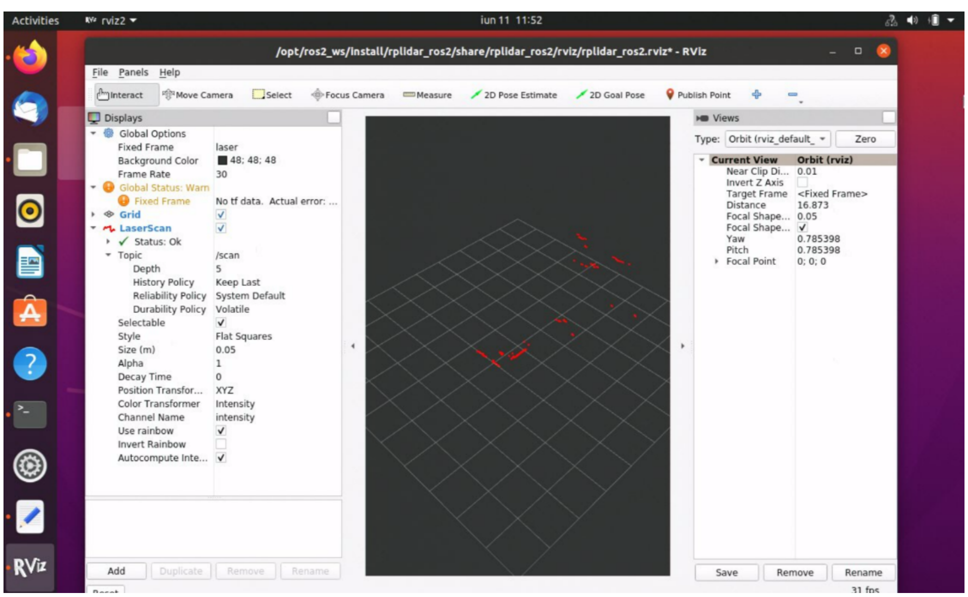
```
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
```

   c. Start the Docker* container for the Slamtec* RPLIDAR:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/01_amr/amr-sdk.all.yml
run rplidar bash
```

   d. Start the Slamtec* RPLIDAR ROS2 node:

```
export ROS_DOMAIN_ID=33
ros2 launch rplidar_ros2 view_rplidar_a3_launch.py &
```



NOTE Use a random number that is unique to your network for the `ROS_DOMAIN_ID`. This will ensure you share the correct topics between Docker images.

4. Start the Cartographer Docker container and start the Cartographer node.

a. Open a new terminal on the same system:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_containers
```

b. Prepare the environment:

```
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
```

c. Open the Cartographer Docker image:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/01_amr/amr-sdk.all.yml
run cartographer bash
```

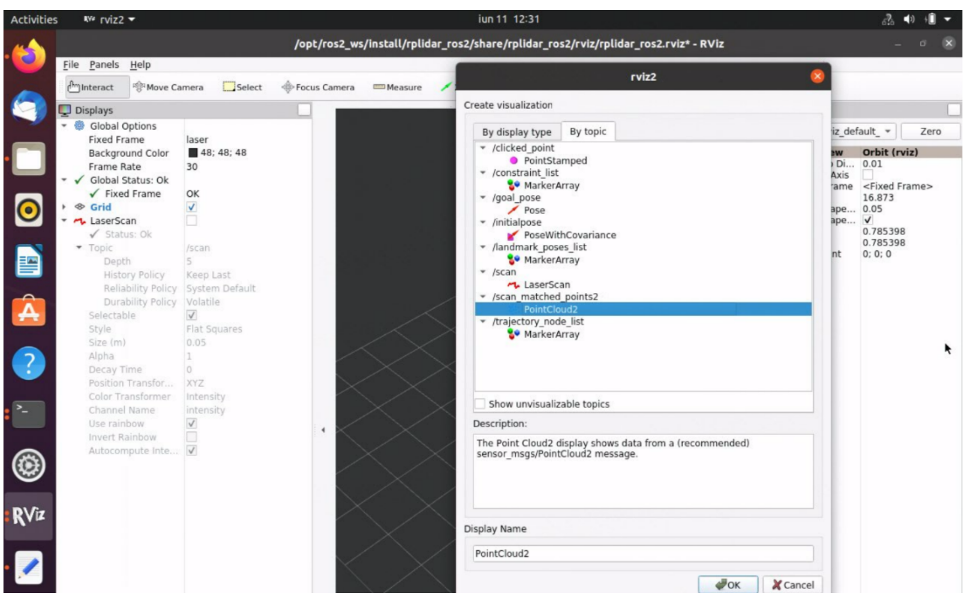
d. Open the Cartographer Docker node:

```
export ROS_DOMAIN_ID=33
source /ros_entrypoint.sh
ros2 run cartographer_ros cartographer_node -configuration_directory /opt/ros/foxy/share/
cartographer/configuration_files/ -configuration_basename rplidar_a3.lua
```

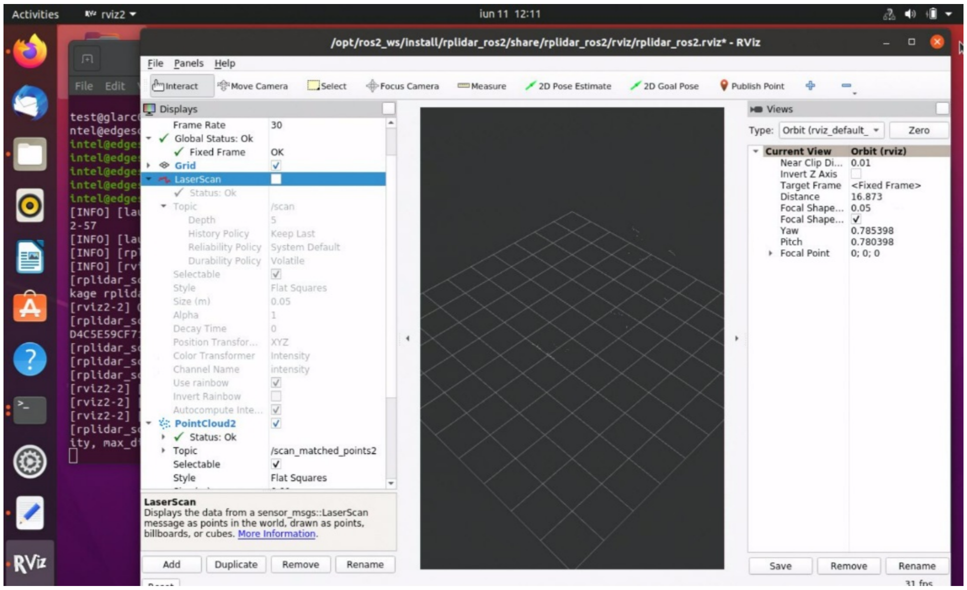
NOTE Use the same number for `ROS_DOMAIN_ID` as the one used for the LIDAR container. This will allow the two Docker images to share ros2 topics.

5. Go to rviz2 that is already open and uncheck **LaserScan** on the left side.

6. Click **Add** in lower left corner, click **By topic** and add **PointCloud2** from the `/scan_matched_points2` topic.



The rviz2 window will look similar to the screenshot below.



Sick NanoScan3 Safety LIDAR

NOTE This tutorial demonstrates how Sick NanoScan3 works inside Edge Insights for Autonomous Mobile Robots. Sick NanoScan3 is a safety laser scanner that needs more configuration to be used in a production environment. Check the Sick website for more details.

1. Connect a Sick NanoScan3 2D LIDAR to your system.

   Hardware setup and configuration can be found on the Sick website.
2. Get the Sick NanoScan3 IP and the host's IP.

   This information can be found when configuring the Sick NanoScan using the Safety Designer, in the *Networking* chapter.
3. Run the Sample Application.

   a. Go to the `AMR_containers` folder:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_containers
```

   b. Prepare the `docker_compose` environment:

```
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
```

   c. Start the Docker* container for the Sick NanoScan3:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/01_amr/amr-sdk.all.yml
run sick-nanoscan bash
```

   d. Add the Sick NanoScan3 IP and host IP in the configuration file:

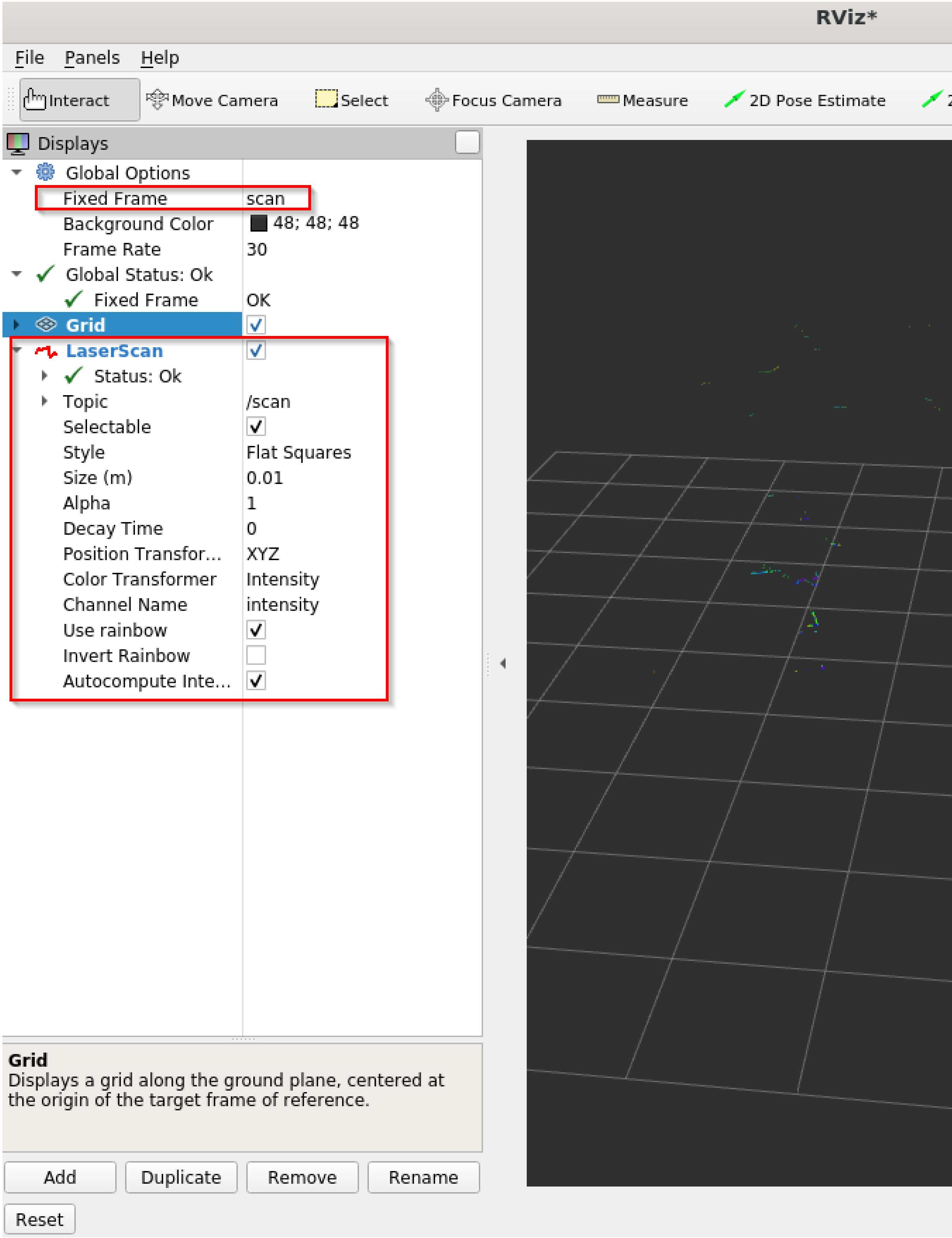
```
sudo vim /opt/ros/foxy/share/sick_safetyscanners2/launch/sick_safetyscanners2_launch.py
```

   Put the Sick NanoScan3 IP in the `sensor_ip` parameter and the host's IP in the `host_ip` parameter.

   e. Start the Sick NanoScan3 ROS2 node:

```
export ROS_DOMAIN_ID=33
ros2 launch sick_safetyscanners2 sick_safetyscanners2_launch.py &
rviz2 &
```

In rviz2, add the LaserScan topic (**Add> By Topic> scan/LaserScan**) and change **Fixed Frame** to **scan**:


RViz*
File
Panels
Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
Displays
Global Options
Fixed Frame scan
Background Color 48; 48; 48
Frame Rate 30
Global Status: Ok
Fixed Frame OK
Grid
LaserScan
Status: Ok
Topic /scan
Selectable
Style Flat Squares
Size (m) 0.01
Alpha 1
Decay Time 0
Position Transfor... XYZ
Color Transformer Intensity
Channel Name intensity
Use rainbow
Invert Rainbow
Autocompute Inte...
Grid
Displays a grid along the ground plane, centered at the origin of the target frame of reference.
Add
Duplicate
Remove
Rename
Reset

4. Start the Cartographer Docker container and start the Cartographer node.

a. Open a new terminal on the same system:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_containers
```

b. Prepare the environment:

```
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
```

c. Open the Cartographer Docker image:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/01_amr/amr-sdk.all.yml
run  cartographer bash
```

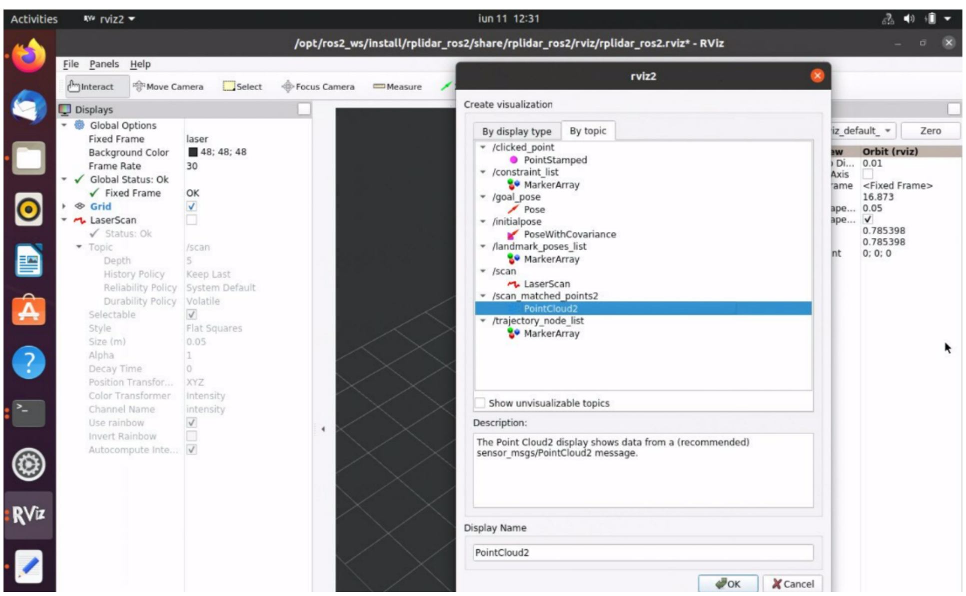
d. Open the Cartographer Docker node:

```
export ROS_DOMAIN_ID=33
source /ros_entrypoint.sh
ros2 run cartographer_ros cartographer_node -configuration_directory /opt/ros/foxy/share/
cartographer/configuration_files/ -configuration_basename sick_nanoscan.lua
```

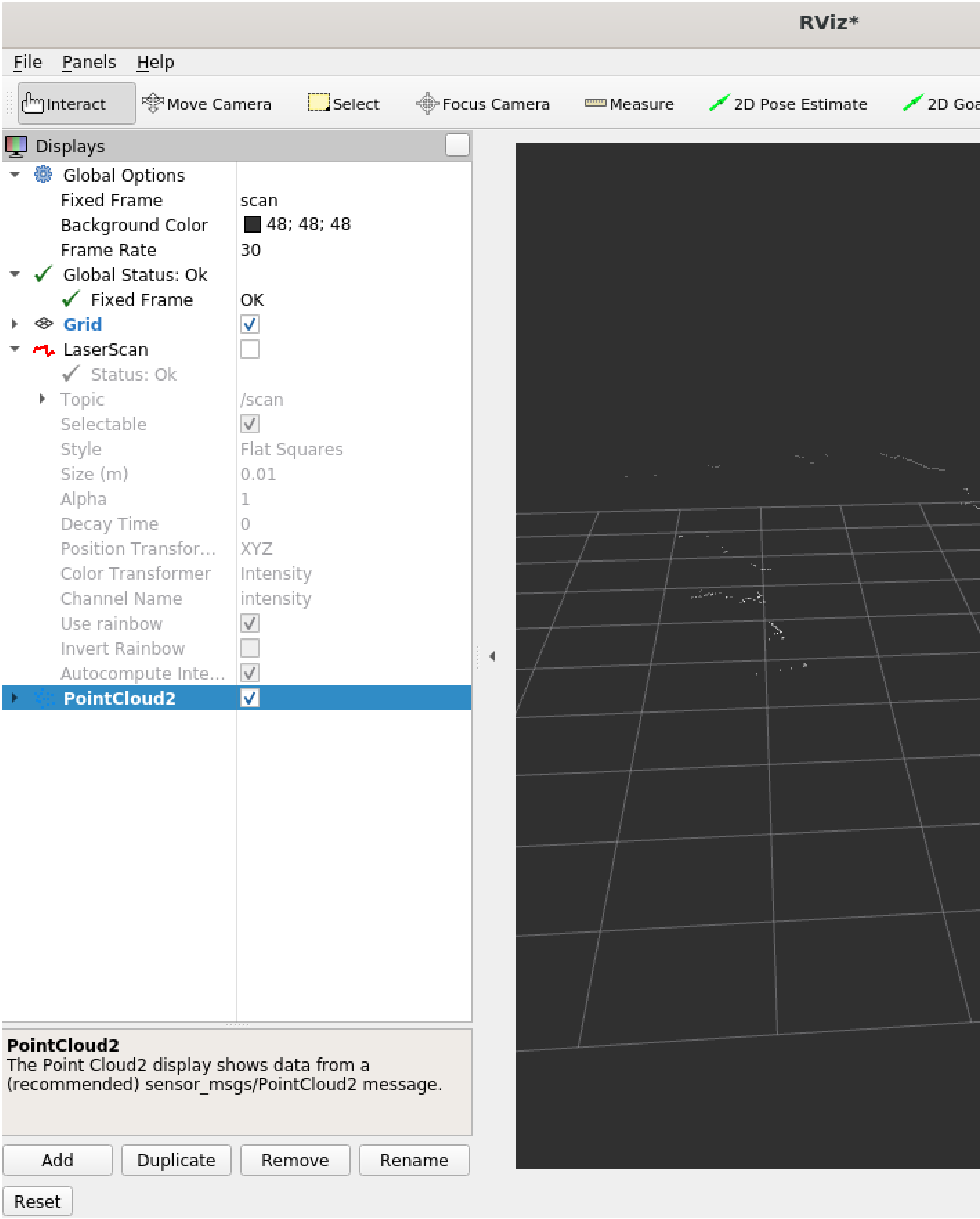
NOTE Use the same number for `ROS_DOMAIN_ID` as the one used for the LIDAR container. This will allow the two Docker images to share ros2 topics.

5. Go to rviz2 that is already open and uncheck **LaserScan** on the left side.

6. Click **Add** in lower left corner, click **By topic** and add **PointCloud2** from the `/scan_matched_points2` topic.



The rviz2 window will look similar to the screenshot below.


RViz*
File
Panels
Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Goa
Displays
Global Options
Fixed Frame
scan
Background Color
48; 48; 48
Frame Rate
30
Global Status: Ok
Fixed Frame
OK
Grid
LaserScan
Status: Ok
Topic
/scan
Selectable
Style
Flat Squares
Size (m)
0.01
Alpha
1
Decay Time
0
Position Transfor...
XYZ
Color Transformer
Intensity
Channel Name
intensity
Use rainbow
Invert Rainbow
Autocompute Inte...
PointCloud2
PointCloud2
The Point Cloud2 display shows data from a (recommended) sensor_msgs/PointCloud2 message.
Add
Duplicate
Remove
Rename
Reset

Troubleshooting

For general robot issues, go to: Troubleshooting for Robot Tutorials.

Run FastMapping Algorithm

Run the Sample Application

1. Go to the `AMR_containers` folder:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_containers
```

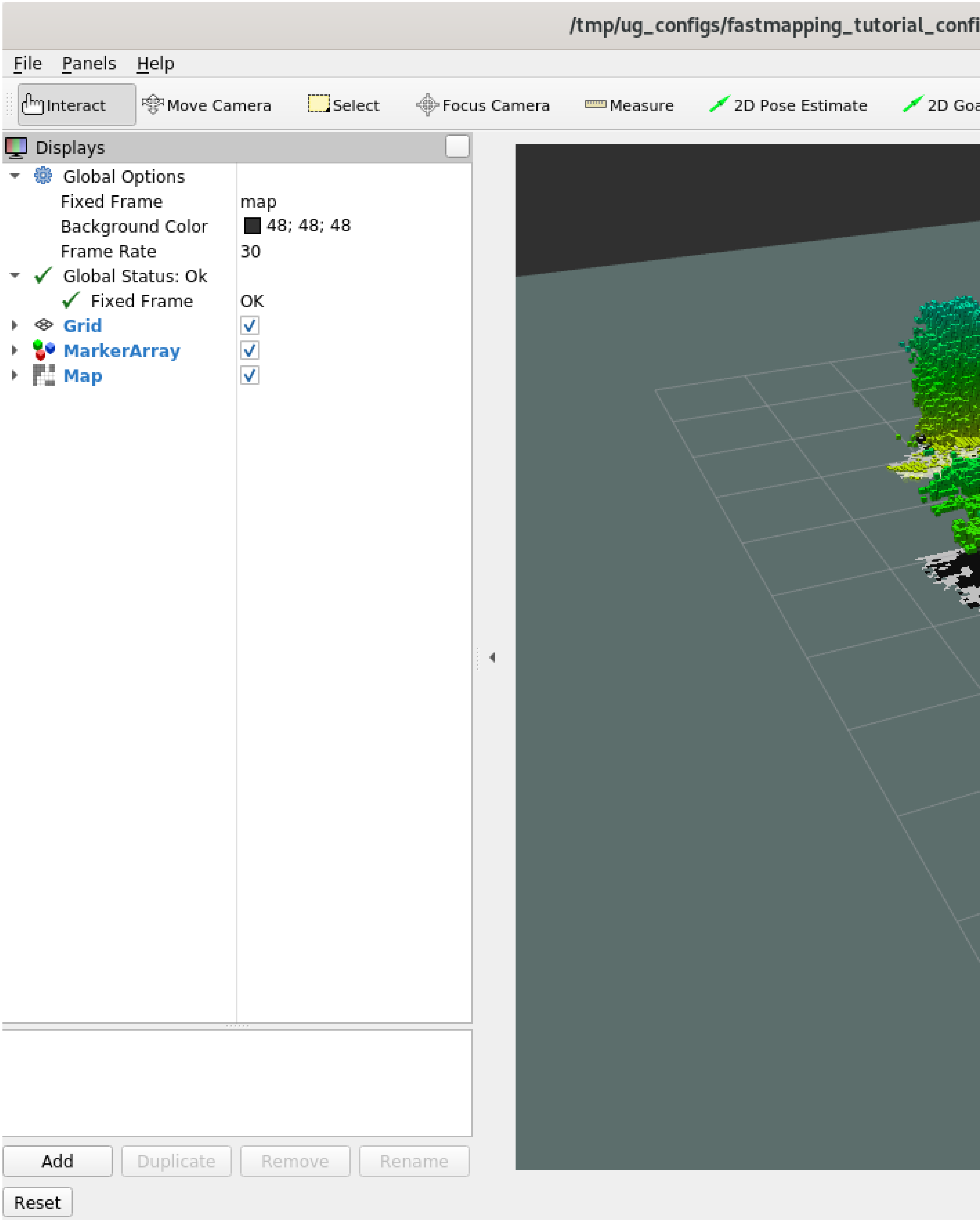
2. Prepare the environment setup:

```
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
export CONTAINER_BASE_PATH=`pwd`
export ROS_DOMAIN_ID=17
# If the bags were not extracted before do it now
unzip 01_docker_sdk_env/docker_compose/06_bags.zip  -d 01_docker_sdk_env/docker_compose/
```

3. Run the FastMapping Algorithm using a bag of a robot spinning:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
fastmapping.demo.yml up
```

Expected output:


/tmp/ug_configs/fastmapping_tutorial_confi
File
Panels
Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Goa
Displays
Global Options
Fixed Frame
map
Background Color
48; 48; 48
Frame Rate
30
Global Status: Ok
Fixed Frame
OK
Grid
MarkerArray
Map
Add
Duplicate
Remove
Rename
Reset

4. To close this, do one of the following:

- Type `Ctrl-c` in the terminal where you did the up command.
- Run this command in another terminal:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
fastmapping.demo.yml down
```

Troubleshooting

For general robot issues, go to: Troubleshooting for Robot Tutorials.

Run ROS 2 Navigation Sample Applications Using the ITS Path Planner Plugin in a Docker* Container

The ITS Plugin for the ROS 2 Navigation 2 application plugin is a global path planner module that is based on Intelligent sampling and Two-way Search (ITS). It does not support continuous replanning.

Prerequisites: Use a simple behavior tree with a compute path to pose and a follow path.

ITS planner inputs:

- global 2D costmap (`nav2_costmap_2d::Costmap2D`)
- start and goal pose (`geometry_msgs::msg::PoseStamped`)

ITS planner outputs: 2D waypoints of the path

Path planning steps summary:

1. The ITS planner converts the 2D costmap to either a Probabilistic Road Map (PRM) or a Deterministic Road Map (DRM).
2. The generated roadmap is saved as a txt file which can be reused for multiple inquiries.
3. The ITS planner conducts a two-way search to find a path from the source to the destination. Either the smoothing filter or a catmull spline interpolation can be used to create a smooth and continuous path. The generated smooth path is in the form of a ROS 2 navigation message type (`nav_msgs::msg`).

For customization options, see ITS Path Planner Plugin Customization.

Run the ROS 2 Navigation Sample Application Using ITS Path Planner

1. Go to the `AMR_containers` folder:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_containers
```

2. Prepare the environment setup:

```
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
export CONTAINER_BASE_PATH=<edge_insights_for_amr_path>/
Edge_Insights_for_Autonomous_Mobile_Robots_*/AMR_containers
```

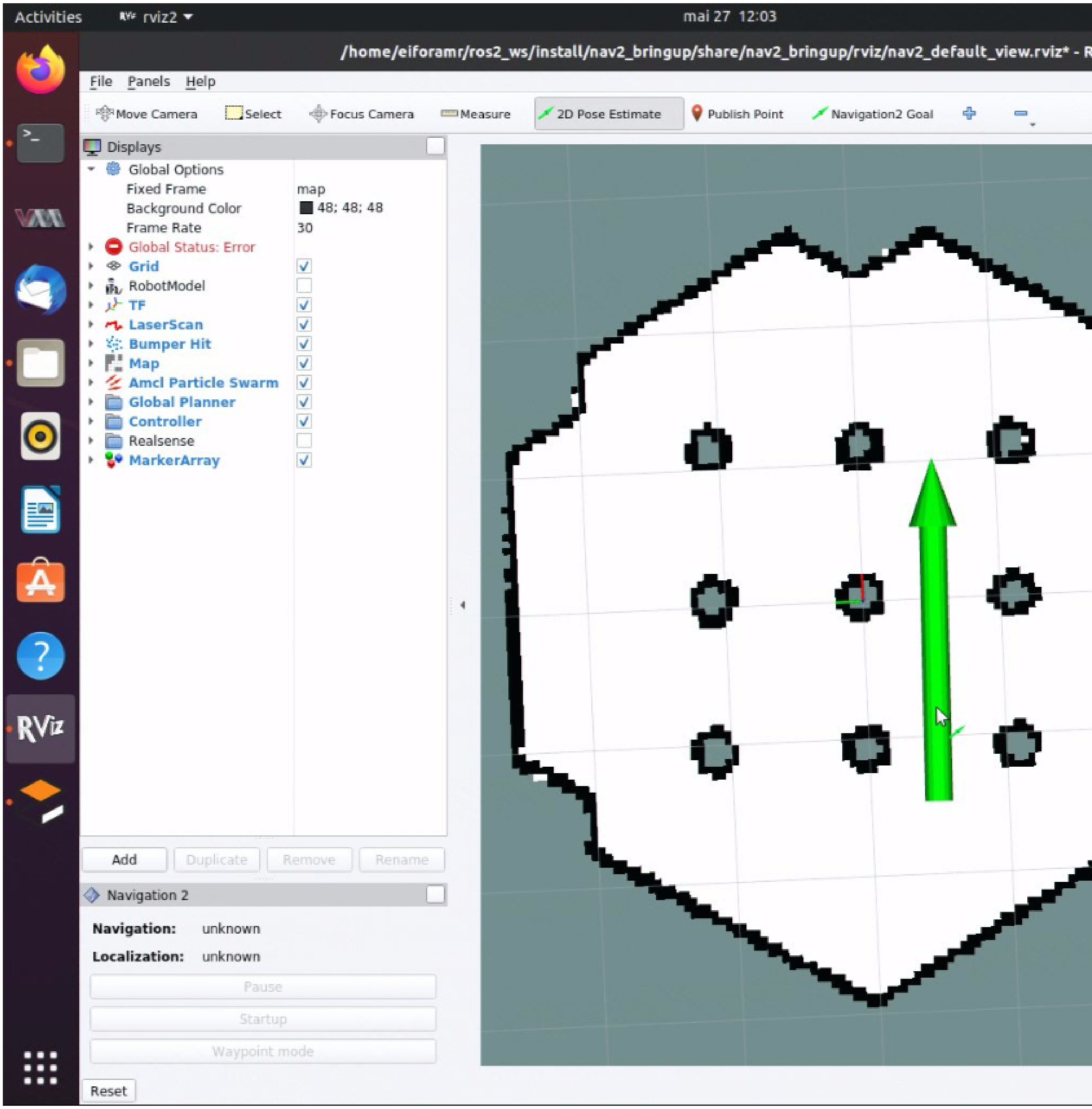
3. Start the ROS 2 navigation sample application using the turtlebot3 Gazebo* simulation:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
its_path_planner.tutorial.yml up
```

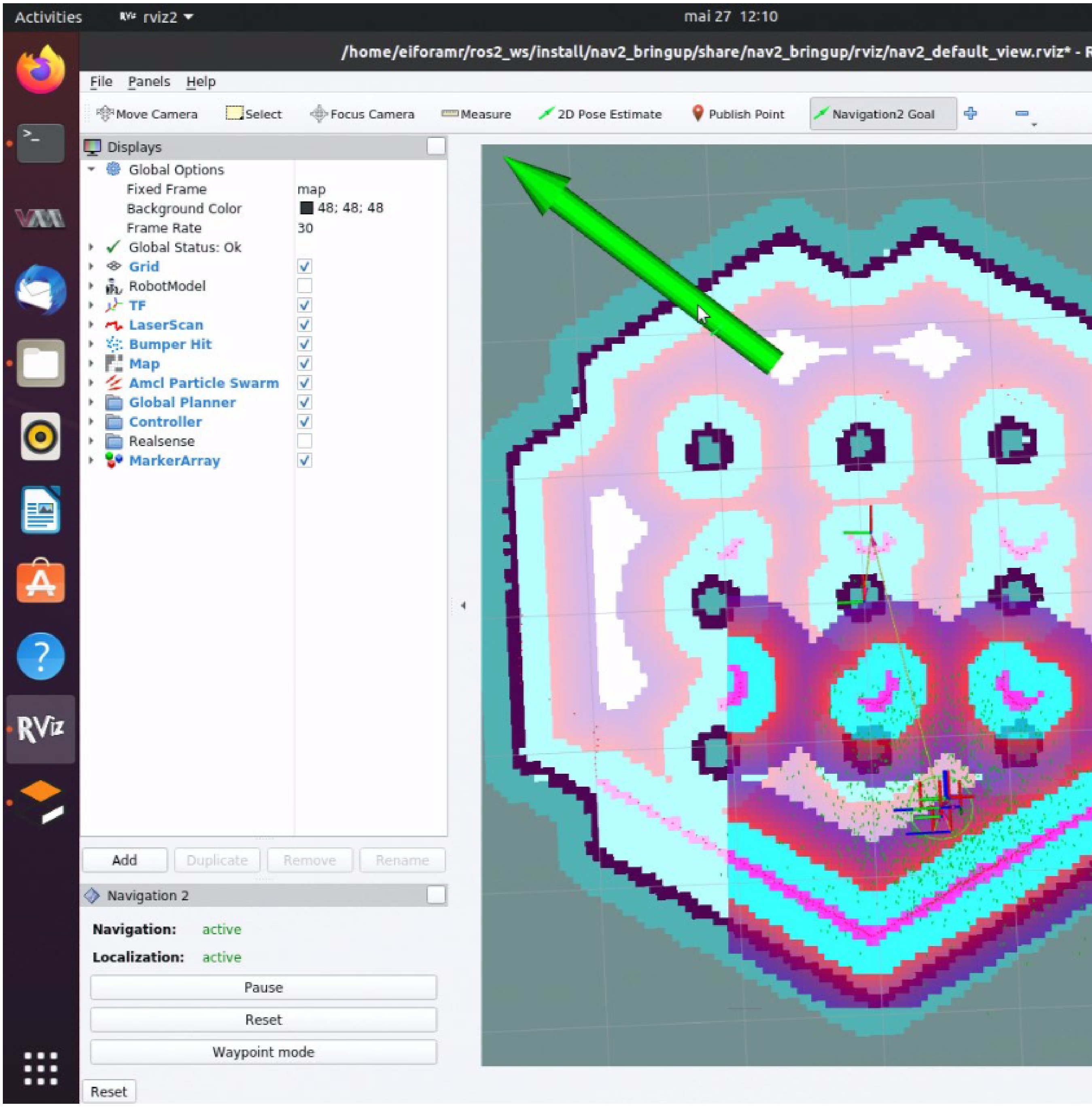
> **NOTE** The above command opens Gazebo* and rviz2 applications. Gazebo* takes a longer time to open (up to a minute) depending on the host’s capabilities. Both applications contain the simulated waffle map, and a simulated robot. Initially, the applications are opened in the background, but you can bring them into the foreground, side-by-side, for a better visual.

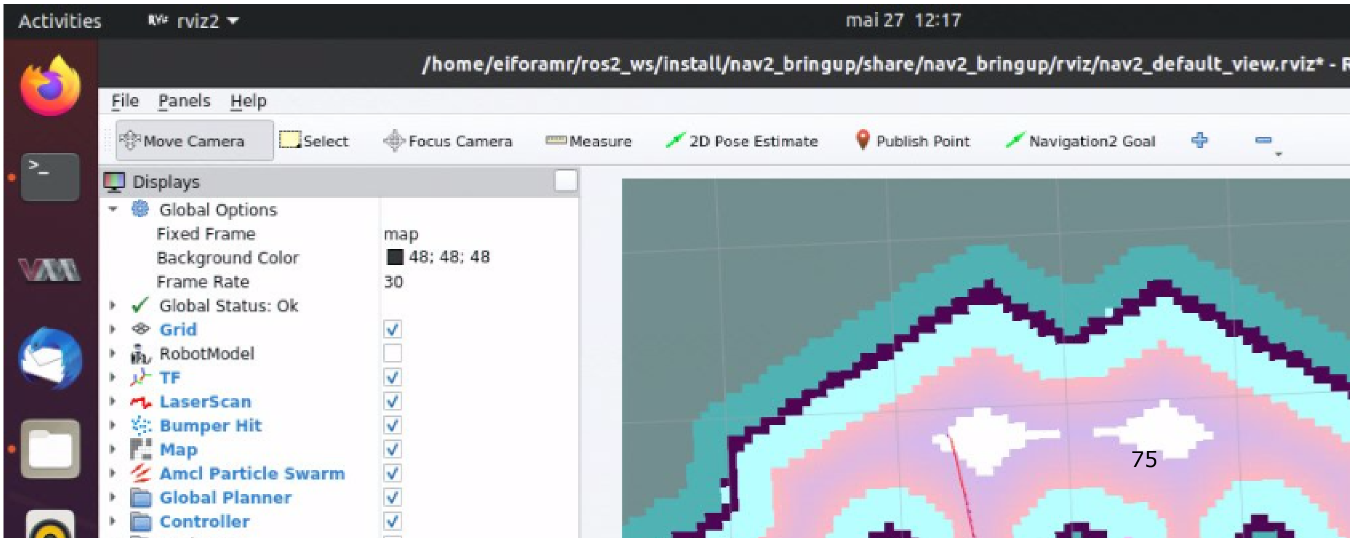
a. Set the robot **2D Pose Estimate** in rviz2:

a. Set the initial robot pose by pressing **2D Pose Estimate** in rviz2.
b. At the robot estimated location, down-click inside the 2D map. For reference, use the robot pose as it appears in Gazebo*.
c. Set the orientation by dragging forward from the down-click. This also enables ROS 2 navigation.



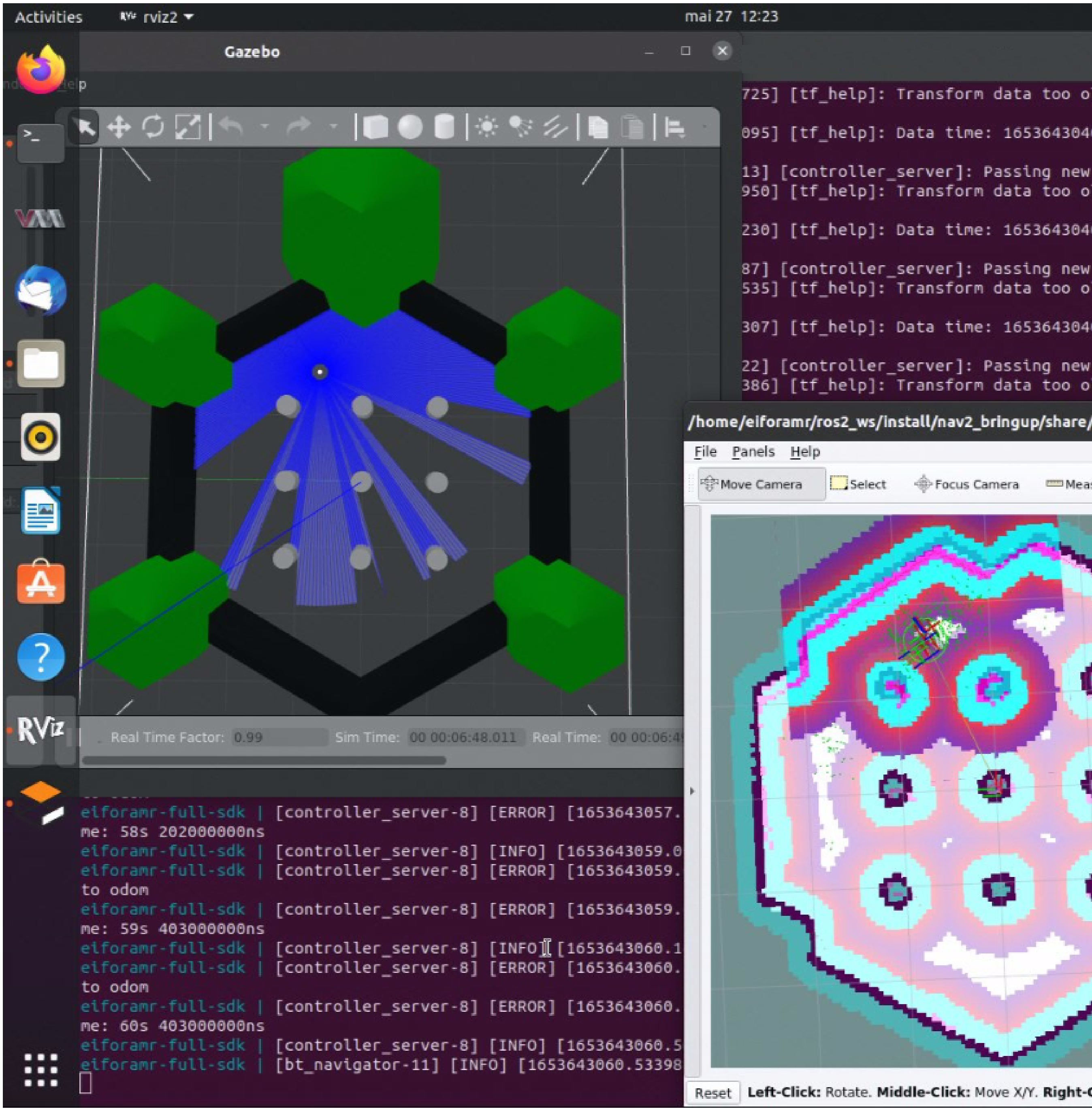
b. In rviz2, press **Navigation2 Goal**, and choose a destination for the robot. This calls the behavioral tree navigator to go to that goal through an action server.


Activities
rviz2
mai 27 12:10
/home/eiforamr/ros2_ws/install/nav2_bringup/share/nav2_bringup/rviz/nav2_default_view.rviz* - R
File Panels Help
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
Publish Point
Navigation2 Goal
Displays
Global Options
Fixed Frame map
Background Color 48; 48; 48
Frame Rate 30
Global Status: Ok
Grid
RobotModel
TF
LaserScan
Bumper Hit
Map
Amcl Particle Swarm
Global Planner
Controller
Realsense
MarkerArray
Add
Duplicate
Remove
Rename
Navigation 2
Navigation: active
Localization: active
Pause
Reset
Waypoint mode
Reset


Activities
rviz2
mai 27 12:17
/home/eiforamr/ros2_ws/install/nav2_bringup/share/nav2_bringup/rviz/nav2_default_view.rviz* - R
File Panels Help
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
Publish Point
Navigation2 Goal
Displays
Global Options
Fixed Frame map
Background Color 48; 48; 48
Frame Rate 30
Global Status: Ok
Grid
RobotModel
TF
LaserScan
Bumper Hit
Map
Amcl Particle Swarm
Global Planner
Controller

Expected result: The robot moves along the path generated to its new destination.

c. Set new destinations for the robot, one at a time.



d. To close this, do one of the following:

- Type `Ctrl-c` in the terminal where you did the up command.
- Run this command in another terminal:

```
CHOOSE_USER=eiforamr 01_docker_sdk_env/docker_compose/05_tutorials/its_path_planner.tutorial.yml
down
```

Troubleshooting

For general robot issues, go to Troubleshooting for Robot Tutorials.

ITS Path Planner Plugin Customization

The ROS 2 navigation bringup application is started using the turtlebot3 Gazebo* simulation, and it receives as input parameter its_nav2_params.yaml.

Check the code snippet from **01_docker_sdk_env/docker_compose/05_tutorials/its_path_planner.tutorial.yml**:

```
export TURTLEBOT3_MODEL=waffle
export GAZEBO_MODEL_PATH=$GAZEBO_MODEL_PATH:/home/eiforamr/ros2_ws/install/turtlebot3_gazebo/
share/turtlebot3_gazebo/models/
ros2 launch nav2_bringup tb3_simulation_launch.py params_file:=${CONTAINER_BASE_PATH}/
01_docker_sdk_env/docker_compose/05_tutorials/param/its_nav2_params.yaml
```

To use the ITS path planner plugin, the following parameters are added in its_nav2_params.yaml:

```
planner_server:
  ros__parameters:
    expected_planner_frequency: 0.01
    use_sim_time: True
    planner_plugins: ["GridBased"]
    GridBased:
      plugin: "its_planner/ITSPlanner"
      interpolation_resolution: 0.05
      catmull_spline: False
      smoothing_window: 15
      buffer_size: 10
      build_road_map_once: True
      min_samples: 250
      roadmap: "PROBABLISTIC"
      w: 32
      h: 32
      n: 2
```

ITS Path Planner Plugin Parameters

```
catmull_spline:
```

If true, the generated path from the ITS is interpolated with the catmull spline method; otherwise, a smoothing filter is used to smooth the path.

```
smoothing_window:
```

The window size for the smoothing filter (The unit is the grid size.)

```
buffer_size:
```

During roadmap generation, the samples are generated away from obstacles. The buffer size dictates how far away from obstacles the roadmap samples should be.

```
build_road_map_once:
```

If true, the roadmap is loaded from the saved file; otherwise, a new roadmap is generated.

```
min_samples:
```

The minimum number of samples required to generate the roadmap

```
roadmap:
```

Either PROBABILISTIC or DETERMINISTIC

```
w:
```

The width of the window for intelligent sampling

```
h:
```

The height of the window for intelligent sampling

```
n:
```

The minimum number of samples that is required in an area defined by `w` and `h`

You can modify these values by editing the file below, at lines 251-267:

```
01_docker_sdk_env/docker_compose/05_tutorials/param/its_nav2_params.yaml
```

Run the Edge Insights for Autonomous Mobile Robots Container in KVM Guest

Run the Sample Application

1. Check if your CPU supports hardware virtualization using the following command. Output greater than zero indicates support is present:

```
egrep -c '(vmx|svm)' /proc/cpuinfo
```

> **NOTE** If the output is not greater than zero, then reboot your server. Modify BIOS Settings > Security features > Enable Virtualization Technology

2. Check if Kernel-based Virtual Machine (KVM) acceleration can be used with the commands:

```
sudo apt install -y cpu-checker
kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used
```

3. Install KVM in Ubuntu* 20.04 using the following commands:

> **NOTE**
>
> 1. If the PAM configuration page is displayed when installing the following packages, click **Yes**.
> 2. If **Configuring openssh-server** page is displayed when installing the following packages, select **keep the local version currently installed**.

```
sudo apt update
sudo apt install -y qemu qemu-kvm libvirt-daemon libvirt-daemon-system libvirt-clients bridge-
utils virt-manager virtinst openssh-server net-tools
sudo adduser $USER libvirt
sudo adduser $USER kvm


# Check if libvirtd is active
sudo systemctl status libvirtd


# If libvirtd is not active use
sudo systemctl enable --now libvirtd
```

4. Reboot the host:

```
sudo reboot -fn
```

After the reboot, verify that libvirtd is active.

```
sudo systemctl status libvirtd
# If libvirtd is not active use
sudo systemctl enable --now libvirtd
```

5. Create the KVM bridge:

   a. Create a new bridge XML file:

```
vim br_kvm.xml
```

   b. Add bridge details to `br_kvm.xml`:

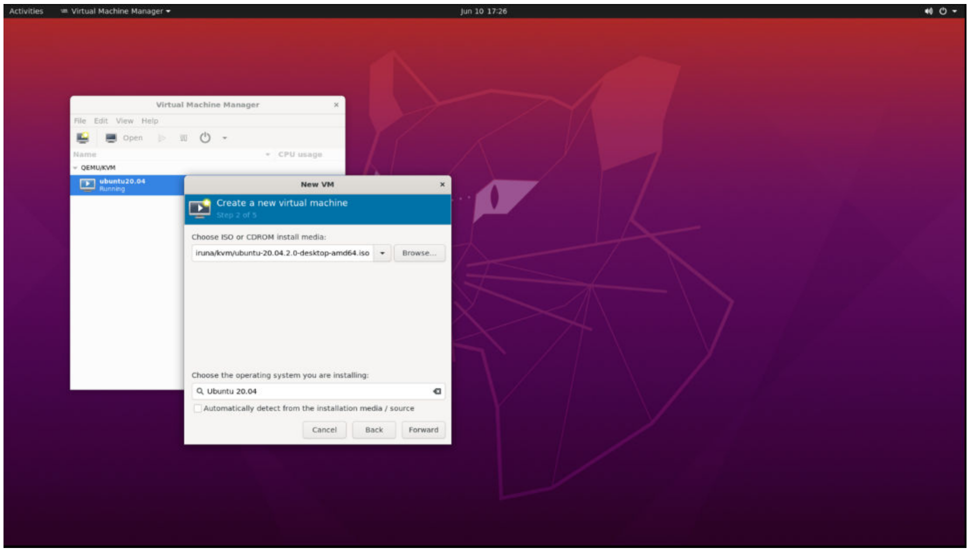
```
<network>
   <name>br_kvm</name>
   <forward mode='nat'>
     <nat>
       <port start='1024' end='65535'/>
     </nat>
   </forward>
   <bridge name='br10' stp='on' delay='0'/>
   <ip address='192.168.124.1' netmask='255.255.255.0'>
     <dhcp>
       <range start='192.168.124.50' end='192.168.124.200'/>
     </dhcp>
   </ip>
 </network>
```

   c. Define and start `br_kvm` network using the following commands:

```
sudo virsh net-define br_kvm.xml
sudo virsh net-start br_kvm
sudo virsh net-autostart br_kvm


# Check if autostart is enabled for br_kvm
sudo virsh net-list --all


Name      State      Autostart   Persistent
 br_kvm      active     yes         yes
 default   active     yes         yes


# Confirm bridge creation and IP address
ip addr show dev br10
13: br10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen
1000
    link/ether 52:54:00:21:ff:4f brd ff:ff:ff:ff:ff:ff
    inet 192.168.124.1/24 brd 192.168.124.255 scope global br10
       valid_lft forever preferred_lft forever
```

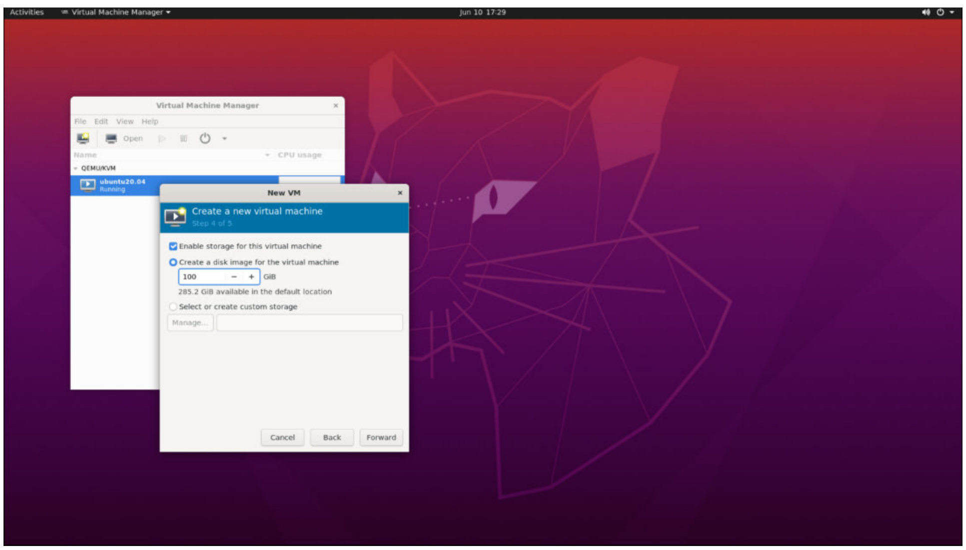
6. Create KVM with Ubuntu 20.04.

   a. Download the Ubuntu 20.04 desktop .iso image from **<releases.ubuntu.com/20.04/>**.
   b. Open virt-manager. Click on **File** > **New VM >** Select **Local install media (ISO image or CDROM)**. Choose the .iso file downloaded and choose the operating system to be installed.

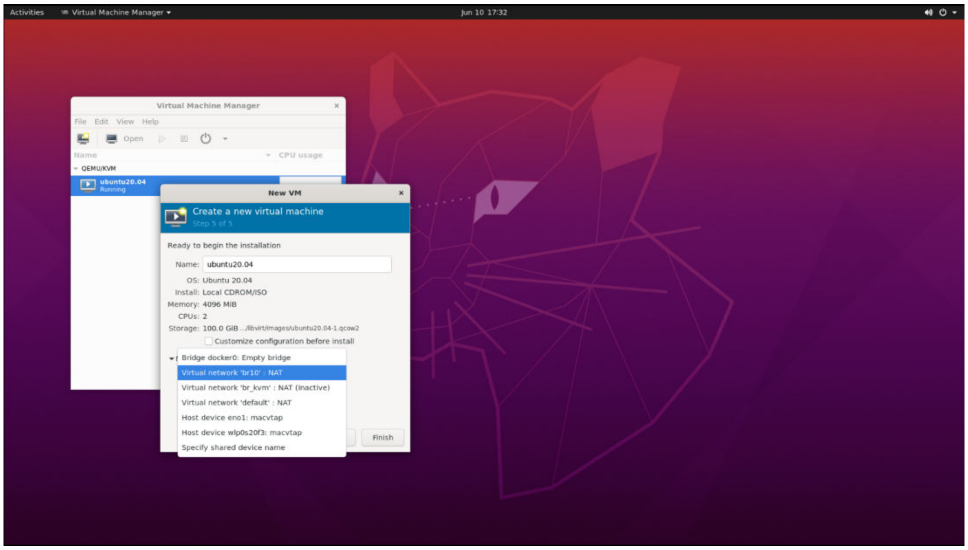


c. Select the CPUs and Memory. For example: Memory: 4096, CPUs: 2

d. Add minimum 100 GB for the virtual machine storage.

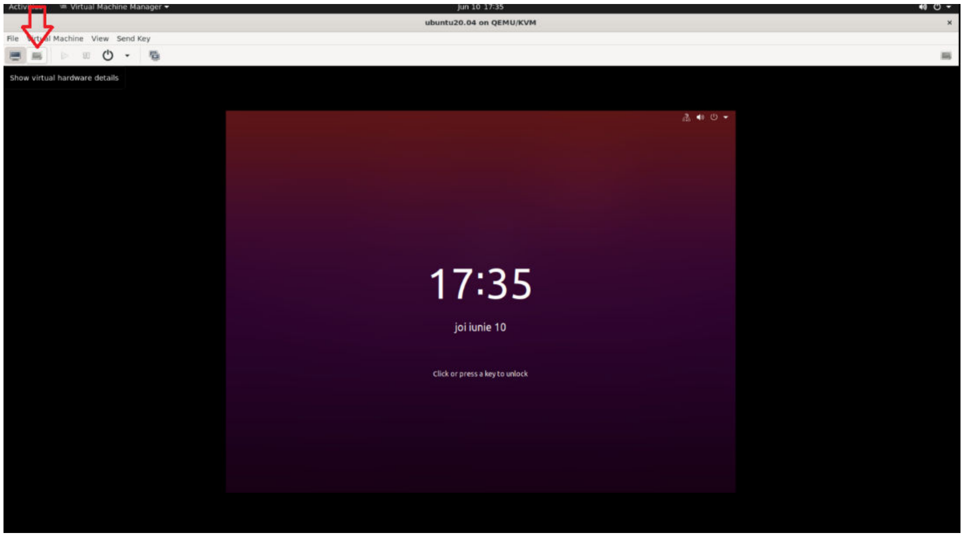


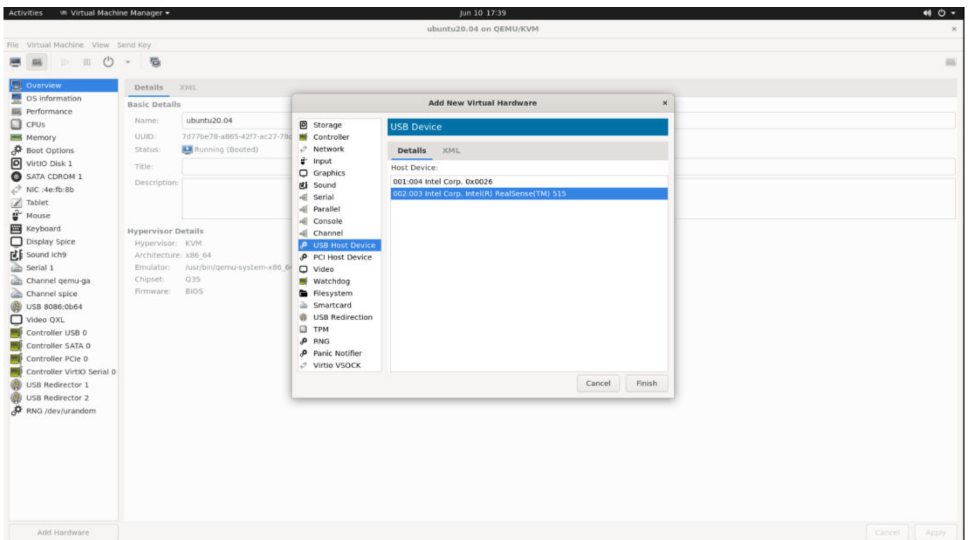
e. Select the Virtual Network created above for Network Preferences.



f. Click **Finish** and start the Ubuntu 20.04 installation.

7. After Ubuntu Installation, add the Intel® RealSense™ camera to the VM by clicking on **Show virtual hardware details** in Virt Manager > **Add Hardware** > **Select USB Host Device** and Select the RealSense Camera.

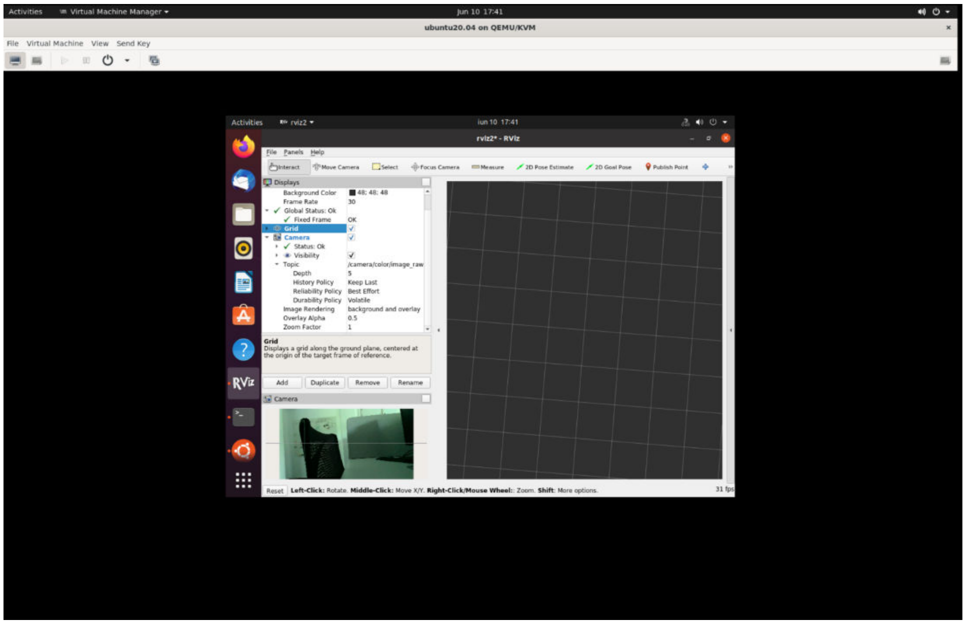




8. Install Edge Insights for Autonomous Mobile Robots using the steps in Get Started Guide for Robots.

NOTE If your system is behind a proxy, you must configure the proxy settings.

9. Run the Intel® RealSense™ ROS 2 sample application inside the Docker container using the steps from Run an Intel® RealSense™ ROS 2 Sample Application in Docker* Container.



Troubleshooting

If the following error is encountered:

```
Error connecting to graphical console:
Error opening Spice console, SpiceClientGtk missing
```

Install `gir1.2-spiceclientgtk-3.0` with the command:

```
sudo apt-get install -y gir1.2-spiceclientgtk-3.0
```

For general robot issues, go to: Troubleshooting for Robot Tutorials.

Run a Collaborative SLAM System

This tutorial tells you how to run a collaborative SLAM system using two ROS 2 bags that simulate two robots exploring the same area.

- The ROS 2 tool rviz2 is used to visualize the two robots, the server, and how the server merges the two local maps of the robots into one common map.
- The output includes the estimated pose of the camera and visualization of the internal map.
- All input and output are in standard ROS 2 formats.

Prerequisites:

- The main input is a camera, either monocular, or stereo, or RGB-D.
- IMU and odometry data are supported as auxiliary inputs.

Run the Sample Application

1. Go to the `AMR_containers` folder:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_containers
```

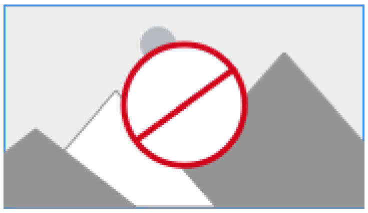
2. Prepare environment setup:

```
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
export CONTAINER_BASE_PATH=`pwd`
export ROS_DOMAIN_ID=17
# If the bags were not extracted before do it now
unzip 01_docker_sdk_env/docker_compose/06_bags.zip  -d 01_docker_sdk_env/docker_compose/
```

3. Run the collaborative SLAM algorithm using two bags simulating two robots going through the same area:

```
CHOOSE_USER=eiforamr docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
cslam.demo.yml up
```

On the server rviz2, both trackers are seen.



In the above figure:

- Red indicates the path robot 1 is taking right now.
- Blue indicates the path robot 2 took.
- Green dots represent points that are known to the server.

Troubleshooting

- Odometry feature `use_odom:=true` does not work with these bags.

  The ros2 bags used in this example do not have the necessary topics recorded for the odometry feature of collaborative slam.

  If the `use_odom:=true` parameter is set, the collab-slam will report errors.
- The bags fail to play.

  The `collab_slam` docker is started with the local user and needs access to the ros2 bags folder.

  Make sure that your local user has read and write access to this path: `<path to edge_insights_for_amr>//Edge_Insights_for_Autonomous_Mobile_Robots_*/AMR_containers/01_docker_sdk_env/docker_compose/06_bags`

  The best method is if your user is the owner of the folder. If the package was installed with sudo, please `chown` the folder to your local user.

For general robot issues, go to: Troubleshooting for Robot Tutorials.

Build New and Custom Docker* Images from the Edge Insights for Autonomous Mobile Robots SDK

This tutorial covers:

- Creating custom Docker* images by adding or removing components in the Docker* files provided in the SDK
- Creating completely new Docker* images by selecting components from the Docker* files provided in the SDK

NOTE Building should be done on a development machine with at least 16 GB of RAM. Building multiple Docker* images, in parallel, on a system with 8 GB of RAM may end in a crash due to lack of system resources.

Create a Customized Docker* Image

1. In the SDK root folder, go to the Docker* environment folder:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_*/AMR_containers/
01_docker_sdk_env/
```

   There are two main Docker* files here:

   - `dockerfile.amr`: For images to be deployed on robots
   - `dockerfile.edge-server`: For images to be deployed on the server

   These Docker* files create multiple Docker* images. The Docker* images to be created are defined in the yaml configuration file, which is in the `docker_compose` folder.

   Also, these Docker* files include many sub-files in the `docker_stages` folder. Each Docker* stage represents a specific component that must be included in one of the Docker* images.

```
# docker_compose and docker_stages folders and dockerfiles are found as below.
ls -a ./
.  ..  artifacts  docker_compose  docker_orchestration  docker_stages  dockerfile.amr
dockerfile.edge-server
```

2. Go to the `docker_stages` folder, and choose the `dockerfile.stage.*` you want to modify:

```
cd docker_stages
```

3. Open the Docker* file from the environment folder in your preferred integrated development environment (IDE), and append component-specific installation instructions in the appropriate place.

   The following is an example of appending the Gazebo* application in `dockerfile.stage.realsense`.

   Give appropriate permissions:

```
chmod 777 dockerfile.stage.realsense
```

```
... <original code from dockerfile.stage.realsense>


################################# Gazebo app START ####################################
RUN sh -c 'echo "deb http://packages.osrfoundation.org/gazebo/ubuntu-stable `lsb_release -cs`
main" > /etc/apt/sources.list.d/gazebo-stable.list'   \
    && wget https://packages.osrfoundation.org/gazebo.key -O - |  apt-key add
-                                                                      \
    && apt-get update && DEBIAN_FRONTEND=noninteractive apt-get install --no-install-recommends -
q -y                                               \
       ros-${ROS_DISTRO}-gazebo-ros-
pkgs
              \
    && rm -rf /var/lib/apt/lists/*
################################# Gazebo app END ####################################
```

4. Build the Docker* image with the modified Docker* file:

   Choose the appropriate Docker* Compose target from the `docker_compose` folder, so that your particular target Docker* image is built.

For example, to build Intel® RealSense™:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_202*
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
docker-compose -f 01_docker_sdk_env/docker_compose/01_amr/amr-sdk.all.yml build realsense
```

NOTE Building should be done on a development machine with at least 16 GB of RAM. Building multiple Docker* images, in parallel, on a system with 8 GB of RAM may end in a crash due to lack of system resources.
Building on the People's Republic of China (PRC) network may result in multiple issues. See the Troubleshooting section for more details.

5. To see the details of the built images:

```
docker images | grep -i amr-
docker images | grep -i edge-server-
```

6. Run the required, newly-installed application from within the container (see Run a ROS 2 Sample Application in the Docker* Container for details).

Create New Docker* Images with Selected Applications from the SDK

In this tutorial, you install an `imaginary` component as a new image and add it to the existing `ros2-foxy-sdk` image.

1. Add a new file called `docker_stages/01_amr/dockerfile.stage.imaginary`.

   Add instructions to install this component into this file using basic Docker* file syntax:

```
# Note: below repo does not exist, it is for demonstration purposes only
WORKDIR ${ROS2_WS}
RUN cd src                                                          \
   && git clone --branch ros2 https://github.com/imaginary.git      \
   && cd imaginary && git checkout <commit_id>                      \
   && source ${ROS_INSTALL_DIR}/setup.bash                          \
   && colcon build --install-base ${ROS2_WS}/install                \
   && rm -rf ${ROS2_WS}/build/* ${ROS2_WS}/src/* ${ROS2_WS}/log/*
```

2. Include this new Docker* file in `dockerile.amr` or `dockerfile.edge-server` at any appropriate location.

   You need to create a separate stage for this new Docker* file so that a separate image can be created using that stage name. For example, append this to `dockerfile.amr`:

```
################################### imaginary stage START ######################################
FROM ros-base AS imaginary
INCLUDE+ docker_stages/01_amr/dockerfile.stage.imaginary
INCLUDE+ docker_stages/01_amr/dockerfile.stage.entrypoint
################################### imaginary stage END ########################################
```

3. If you need to add this new component to other images also, add it inside any stage.

   For example, to add `imaginary` to the `ros2-foxy-sdk` stage:

```
################################ ros2-foxy-SDK stage START ####################################
FROM ros-base AS ros2-foxy-sdk
INCLUDE+ docker_stages/common/dockerfile.stage.tools-dev

####### add new component on appropriate place in this block #######
INCLUDE+ docker_stages/01_amr/dockerfile.stage.imaginary

INCLUDE+ docker_stages/01_amr/dockerfile.stage.vda5050
INCLUDE+ docker_stages/01_amr/dockerfile.stage.imaginary
```

```
INCLUDE+ docker_stages/01_amr/dockerfile.stage.opencv
INCLUDE+ docker_stages/01_amr/dockerfile.stage.rtabmap
INCLUDE+ docker_stages/01_amr/dockerfile.stage.fastmapping
INCLUDE+ docker_stages/01_amr/dockerfile.stage.gazebo
INCLUDE+ docker_stages/01_amr/dockerfile.stage.gstreamer
INCLUDE+ docker_stages/01_amr/dockerfile.stage.kobuki
INCLUDE+ docker_stages/01_amr/dockerfile.stage.nav2
INCLUDE+ docker_stages/01_amr/dockerfile.stage.realsense
INCLUDE+ docker_stages/01_amr/dockerfile.stage.ros-arduino
INCLUDE+ docker_stages/01_amr/dockerfile.stage.ros1-bridge
INCLUDE+ docker_stages/01_amr/dockerfile.stage.rplidar
INCLUDE+ docker_stages/01_amr/dockerfile.stage.turtlebot3
INCLUDE+ docker_stages/01_amr/dockerfile.stage.turtlesim
# simlautions has hard dependency in nav2 (@todo:), so we can not create separate image for
simulations without nav2.
INCLUDE+ docker_stages/01_amr/dockerfile.stage.simulations
INCLUDE+ docker_stages/01_amr/dockerfile.stage.entrypoint
################################# ros2-foxy-SDK stage END ######################################
```

4. Define a new target in the `docker_compose/amr-sdk.all.yml` or `docker_compose/edge-server.all.yml` file:

```
imaginary:
    image: ${REPO_URL}amr-ubuntu2004-ros2-foxy-imaginary:${DOCKER_TAG:-latest}
    container_name: ${CONTAINER_NAME_PREFIX:-amr-sdk-}imaginary
    extends:
       file: ./amr-sdk.all.yml
       service: ros-base
    build:
       target: imaginary
    network_mode: host
    command: ['echo imaginary run finished.']
```

5. Build two Docker* images:

   - amr-ubuntu2004-ros2-foxy-imaginary
   - amr-ubuntu2004-ros2-foxy-sdk

   These images contain the new `imaginary` component.

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_202*
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
docker-compose -f 01_docker_sdk_env/docker_compose/01_amr/amr-sdk.all.yml build imaginary ros2-
foxy-sdk
```

6. To see the details of the built image:

```
docker images | grep -i amr-
docker images | grep -i imaginary
```

7. To verify that the `imaginary` component is part of created Docker* image:

```
docker history amr-ubuntu2004-ros2-foxy-imaginary
```

8. Run the required, newly-installed application from within the container (see Run a ROS 2 Sample Application in the Docker* Container for details).

Troubleshooting

1. Building on the People’s Republic of China (PRC) Open Network.

Building Docker* images on the People's Republic of China (PRC) open network may fail. Intel recommends updating these links with their corresponding PRC mirrors. To do this, go to the `AMR_containers` folder, and update the broken sites with the default or user-defined mirrors.

```
cd Edge_Insights_for_Autonomous_Mobile_Robots_*/AMR_containers
chmod 775 changesources.sh
./changesources.sh -d .
Enter mirror server ('https://example.com' format) or leave empty to use the default value.
Git mirror [https://github.com.cnpmjs.org]:
Apt mirror [http://mirrors.bfsu.edu.cn]:
Pip mirror [https://opentuna.cn/pypi/web/simple/]:
Raw files mirror [https://raw.staticdn.net]:
```

2. Building on a limited resource system (8 GB of RAM or less) can be problematic.

   Perform the following steps to minimize issues:

   a. Save the output in a file instead of printing it, because printing consumes RAM resources.

```
nohup time docker-compose -f 01_docker_sdk_env/docker_compose/01_amr/amr-sdk.all.yml build --
parallel > eiforAMR.txt &
```

NOTE Edge Insights for Autonomous Mobile Robots comes with prebuilt images and building all images is not required. Only do this step if you want to regenerate all images.

   b. For remote connections, use an ssh connection instead of a VNC one as VNC connection consumes resources.
   c. For building multiple Docker* images, do not use the `--parallel` option as it requires more resources.

```
nohup time docker-compose -f 01_docker_sdk_env/docker_compose/01_amr/amr-sdk.all.yml build >
eiforAMR.txt &
```

NOTE Edge Insights for Autonomous Mobile Robots comes with prebuilt images and building all images is not required. Only do this step if you want to regenerate all images.

Troubleshooting for Robot Tutorials

This guide lists common troubleshooting tips for robot tutorials.

Permission Denied Error

For a permission denied error when running a script:

```
$ ./run_interactive_docker.sh eiforamr-full-flavour-sdk:<TAG> eiforamr
bash: ./run_interactive_docker.sh Permission denied
```

Give executable permission to the script:

```
chmod 755 run_interactive_docker.sh
```

DISPLAY Environment Variable Error

For errors related to the `DISPLAY` environment variable when trying to open the Docker container or a GUI application, enter the command:

```
echo $DISPLAY
```

If this variable is empty, it will cause issues when opening applications that need GUI.

The most common solution is to give it the `0:0` value:

```
export DISPLAY= "0:0"
```

If the connection with the system is via VNC, `DISPLAY` should be already set.

If it is not, find out the value of `DISPLAY` set by vncserver and then set the correct value:

For example:

```
ps ax |grep vncserver
   /usr/bin/Xtigervnc :42 -desktop ....
   /usr/bin/perl /usr/bin/vncserver -localhost no -geometry 1920x1000 -depth 24 :42

export DISPLAY= ":42"
```

Use ROS_DOMAIN_ID to Avoid Interference in ROS Messages

A typical method to demonstrate a use case requires you to start a container (or group of containers) and exchange ROS messages between various ROS nodes. However, interference from other ROS nodes can disrupt the whole process. For example, you might receive ROS messages from unknown nodes that are not intended for the demo use case. These other nodes could be on the same host machine or on other host machines within the local network. In this scenario, it can be difficult to debug and resolve the interference.

You can avoid this by declaring `ROS_DOMAIN_ID` as a fixed numeric value per use case, under the following conditions:

- The `ROS_DOMAIN_ID` should be same for all containers launched for a particular use case.
- The `ROS_DOMAIN_ID` should be an integer between 0 and 101.
- After launching the container, you can declare it with:

```
export ROS_DOMAIN_ID=<value>
```

For more information, go to: ROS_DOMAIN_ID

To add the `ROS_DOMAIN_ID`, you can choose any of the below options.

1. Add it in the `common.yml` file for all containers:

```
# In file 01_docker_sdk_env/docker_compose/common/common.yml
# ROS_DOMAIN_ID can be added that applies to all use cases
services:
common:
   environment:
   ROS_DOMAIN_ID: <choose ID>
```

2. Add it in the `.env` file for all containers:

```
# In file 01_docker_sdk_env/docker_compose/01_amr/.env
# add below line and provide ROS_DOMAIN_ID
ROS_DOMAIN_ID=<choose ID>
```

3. Add it in the specific yml file for a specific use case for specific targets:

```
# In the below example, ROS_DOMAIN_ID is added in ros-base target
# For any use case where this target is used, the ROS_DOMAIN_ID will be set to the given value.

services:

   ros-base:
      image: ${REPO_URL}amr-ubuntu2004-ros2-foxy-ros-base:${DOCKER_TAG:-latest}
      container_name: ${CONTAINER_NAME_PREFIX:-amr-sdk-}ros-base
      environment:
         ROS_DOMAIN_ID: <choose ID>
```

```
      env_file:
         - ./.env
      extends:
```

4. Add it in the specific yml file in the `command:` section and apply only after launching the containers:

```
# In file 01_docker_sdk_env/docker_compose/05_tutorials/
fleet_mngmnt_with_low_battery.up.tutorial.yml
# In the below example, ROS_DOMAIN_ID is set to 58
# You may change it to any new value as per use case requirement.
services:

battery_bridge:
      image: ${REPO_URL}amr-ubuntu2004-ros2-foxy-battery_bridge:${DOCKER_TAG:-latest}
      container_name: ${CONTAINER_NAME_PREFIX:-amr-sdk-}battery_bridge
      extends:
      file: ../01_amr/amr-sdk.all.yml
      service: ros-base
      volumes:
      - /dev/battery_bridge:/dev/battery_bridge:rw
      build:
      target: battery_bridge
      network_mode: host
      restart: "no"
      command:
      - |
         source ros_entrypoint.sh
         source battery-bridge/src/prebuilt_battery_bridge/local_setup.bash
         export ROS_DOMAIN_ID=58
         sleep 5
         ros2 run battery_pkg battery_bridge
```

5. Add it while running a container using the `run_interactive_docker.sh` script:

```
# by adding env parameter, ROS_DOMAIN_ID can be exported inside container:
./run_interactive_docker.sh <image name> <user> --extra_params "-e ROS_DOMAIN_ID=<choose ID>"
```

NOTE You can use any number between 0 and 101 (inclusive), to set `ROS_DOMAIN_ID`, as long as it is not used by a different ROS system.

Be aware that you can also use these options to modify other environment variables.

System HOME Directory Issues

If your test system uses `$HOME` mounted in remote volumes, for example, in a network file system (NFS), you may encounter the error below when you try to run a Docker* image using the `./run_interactive_docker.sh` script:

```
docker: Error response from daemon: error while creating mount source path '/nfs/site/home/
<user>': mkdir /nfs/site/home/<user>: file exists.
```

To avoid this, before you run a Docker* image, create a new directory in `/tmp` (or any locally mounted volume), and set `$HOME` to the new path:

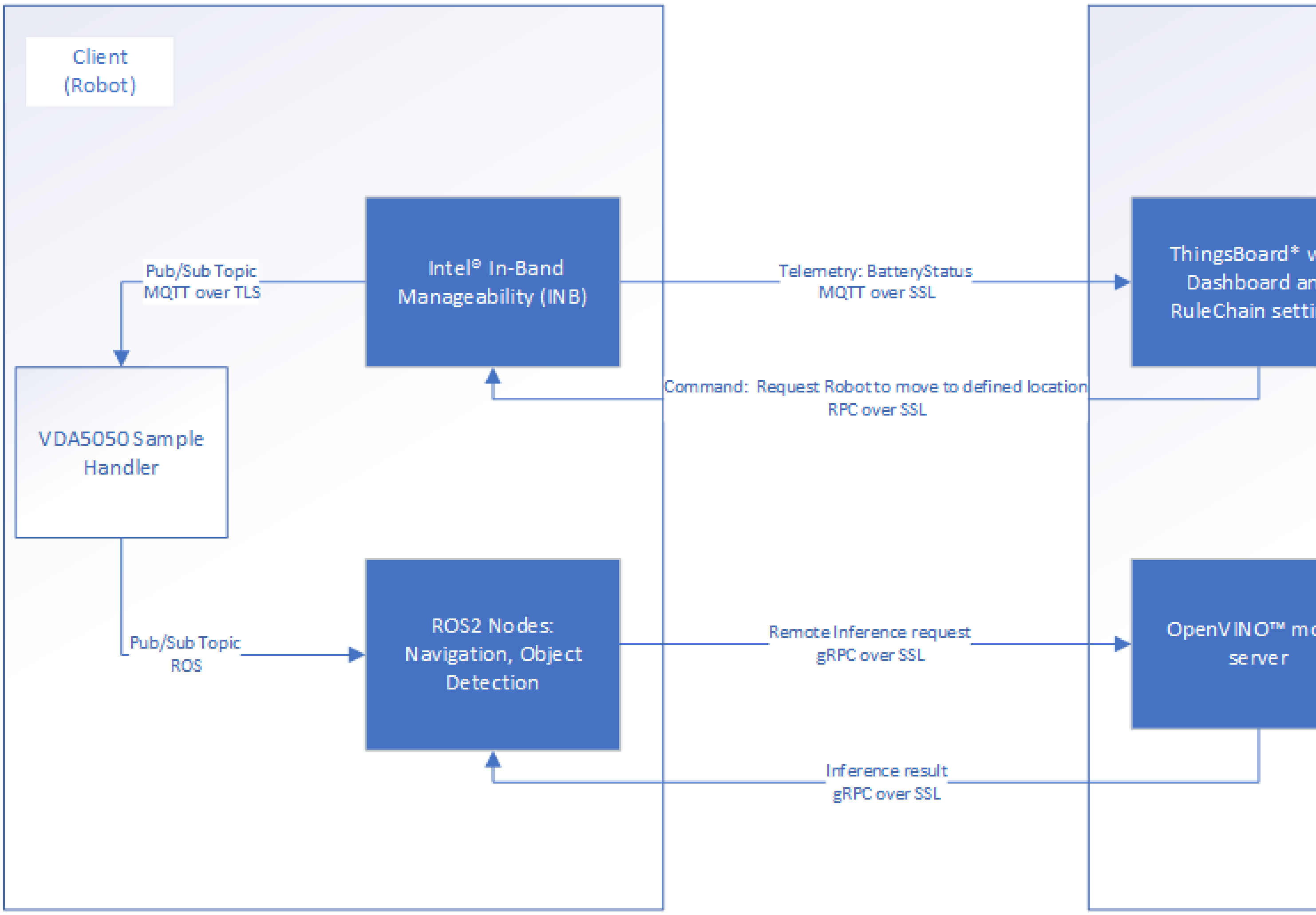
```
mkdir /tmp/tmp_home
export HOME=/tmp/tmp_home
./run_interactive_docker.sh eiforamr-full-flavour-sdk:<release_tag> eiforamr
```

Fleet Tutorials

- Basic Fleet Management
- Device Onboarding End-to-End Use Case
- OTA Updates
- Wandering Application Deployment
- Troubleshooting for Robot Orchestration Tutorials

Basic Fleet Management

The basic fleet management solution consists of server and client architecture. The following diagram presents the architecture, components, and communication between components.



Basic Fleet Management use cases are triggered by battery level:

- Basic Fleet Management use case, commanding a robot to return to a docking station (See the Get Started Guide for Robot Orchestration.)
- Remote Inference use case with a remote inference request to an OpenVINO™ model server

The basic fleet management server is one of the microservices of orchestration, and it is the solution provided by ThingsBoard* (https://thingsboard.io/docs/user-guide/install/docker/).

Using a ThingsBoard* server to connect to the Intel® In-Band Manageability framework (https://github.com/intel/intel-inb-manageability), deployed clients are able to provide fleet management and telemetry. The ThingsBoard* server GUI gives a clear view of the telemetry data with the Intel® In-Band Manageability-tailored dashboard. In addition, rules can be set for configured, validated events to reach fleet management use cases.

The basic fleet management client (which is deployed on robots) consists of Intel® In-Band Manageability, the VDA5050 sample handler, and ROS 2 nodes (can be navigation or object detection purposes). When a subscribed topic is published by Intel® In-Band Manageability, the VDA5050 sample handler processes the VDA5050 complied JSON format and translates it into ROS2 topics to publish.

The VDA5050 complied JSON format message can be conducted in the ThingsBoard* Rule Engine (https://thingsboard.io/docs/user-guide/rule-engine-2-0/re-getting-started/) nodes with configured telemetry message validation and sent via an RPC call node, or it can be sent manually on the GUI.

For the remote inference use case, the requests from ROS 2 node go to OpenVINO™ model server (https://github.com/openvinotoolkit/model_server/tree/main/extras/nginx-mtls-auth) via SSL channel.

- Basic Fleet Management Use Case
- Remote Inference End-to-End Use Case

Basic Fleet Management Use Case

This tutorial tests your install by setting up the fleet management server to guide an EI for AMR to the docking station when its battery reaches the 40% threshold.

You must do all of the sections in this tutorial in the order listed.

- Machine A is the server.
- Machine B is an EI for AMR target that sends data to the server and receives instructions (the robot).
- Machine A and Machine B need to be in the same network.

Prerequisites: The robot and the server are configured as instructed in their Get Started Guides:

- Configuring the client: Get Started Guide for Robots.
- Configuring the server: Get Started Guide for Robot Orchestration.

Build, and Configure the Basic Fleet Management Client

On Machine B

1. Go to the `AMR_containers` folder:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_containers
```

2. Set up the environment variables necessary to run docker-compose commands:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_202*/AMR_containers/
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
```

3. Make sure that the server public key generated in the previous step is copied to the right path in client sources and that it does not contain the default dummy content:

```
cat 01_docker_sdk_env/artifacts/02_edge_server/edge_server_fleet_management/thingsboard.pub.pem
```

4. Generate configuration file:

```
chmod 775 ./01_docker_sdk_env/artifacts/01_amr/amr_fleet_management/turtle_creek_client/
cloud_source
chmod +x ./01_docker_sdk_env/artifacts/01_amr/amr_fleet_management/turtle_creek_client/
fleet_client_prereq.sh
sudo ./01_docker_sdk_env/artifacts/01_amr/amr_fleet_management/turtle_creek_client/
fleet_client_prereq.sh  <Server-IP> <PORT> y 8nwWlQnkdZn5HShvRekx TGL1-i7
```

NOTE Parameters: `<Server-IP><PORT><TLS:Y/N><DeviceToken><product-name-of-robot>`
`<Server-IP>`: Use the Controller IP if Intel® Smart Edge Open Multi-Node is deployed. Use ThingsBoard* server pod's IP if Intel® Smart Edge Open Single-Node is deployed.

`<PORT>`: Use 32767 as the one configured in the Intel® Smart Edge Open playbook.

`<TLS>`: Default use Y to enable TLS connection with ThingsBoard* server.

`<DeviceToken>`: Default now with preconfigured db. New one can be used when a new device profile is created in the ThingsBoard* server.

`<product-name-of-robot>`: Get the product name of the robot by running sudo dmidecode -t system | grep Product

Each time you change the above parameters, do not forget to rerun the `fleet_client_prereq.sh` script.

5. Build the basic fleet management Docker* image:

```
docker-compose -f 01_docker_sdk_env/docker_compose/01_amr/amr-sdk.all.yml build fleet-management
```

6. Install the battery-bridge kernel module:

   When the EI for AMR (machine B) uses an actual battery, the sensor-driver of the robot provides the corresponding driver's ros-interface, which writes battery status into generic ros2-topic interface /sensors/battery_state. However, this information is usually not transmitted to the generic OS interface /sys/class/power_supply. Components that interact with OS directly (for example, Intel® In-Band Manageability), cannot get battery-information from OS.

   To bridge this gap, a ROS component battery-bridge and a kernel module battery-bridge-kernel-module is provided. Using this battery-bridge, battery-status can be transmitted via a kernel module to standard OS interface /sys/class/power_supply.

   Koboki driver and Kobuki-ros-interface is proven to be working with battery-bridge and battery-bridge-kernel-module components.

```
cd components/amr_battery_bridge_kernel_module/src/
chmod a+x module_install.sh
# below command will install battery-bridge-kernel-module
sudo module_install.sh
# to uninstall battery-bridge-kernel-module (if needed)
sudo module_install.sh -u
```

   Known Limitations

   Make sure that **UEFI Secure Boot** is disabled:

   a. Go to the **BIOS** menu.
   b. Open **Boot** > **Secure Boot**.
   c. Disable **Secure Boot**.

   If the battery-bridge-kernel-module cannot be installed, see: Troubleshooting for Robot Orchestration Tutorials, "battery-bridge-kernel-module Install Failure".

Start the Basic Fleet Management Client Deployment

On Machine B

1. Go to the `AMR_containers` folder:

```
cd Edge_Insights_for_Autonomous_Mobile_Robots_*/AMR_containers/
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
export CONTAINER_BASE_PATH=pwd
```

2. Run the Basic Fleet management turtle_creek_client container:

```
CHOOSE_USER=root docker-compose -f 01_docker_sdk_env/docker_compose/01_amr/amr-sdk.all.yml up
amr-fleet-management
```

3. Run the wandering app microservices:

```
CHOOSE_USER=root docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
aaeon_wandering_remote_inference.yml up battery-bridge realsense aaeon-amr-interface ros-base-
camera-tf collab-slam fastmapping nav2 wandering vda5050-ros2-bridge
```

4. If no battery bridge installed, run:

```
CHOOSE_USER=root docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
aaeon_wandering_remote_inference.yml up realsense aaeon-amr-interface ros-base-camera-tf collab-
slam fastmapping nav2 wandering vda5050-ros2-bridge
```

Log into the Basic Fleet Management Server

On Machine A

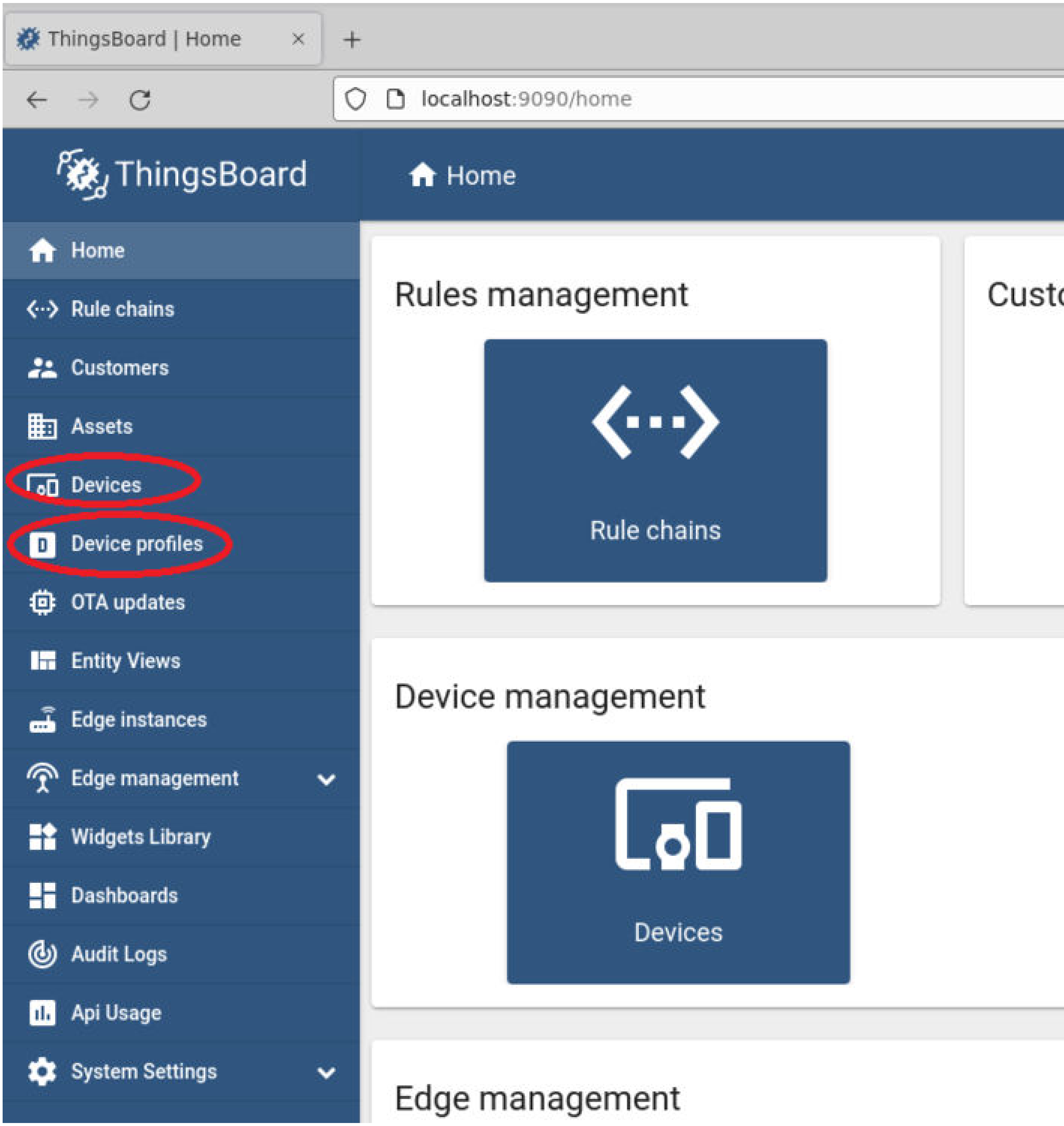
1. Open the Basic Fleet Management Dashboard:

> **NOTE** VNC interferes with the Intel® Smart Edge Open installation. Intel recommends that you open the Basic Fleet Management Dashboard on a different system, as the dashboard is accessible via internet.

```
# Open a browser, use controller IP and open:
<IP Address>:32764
```

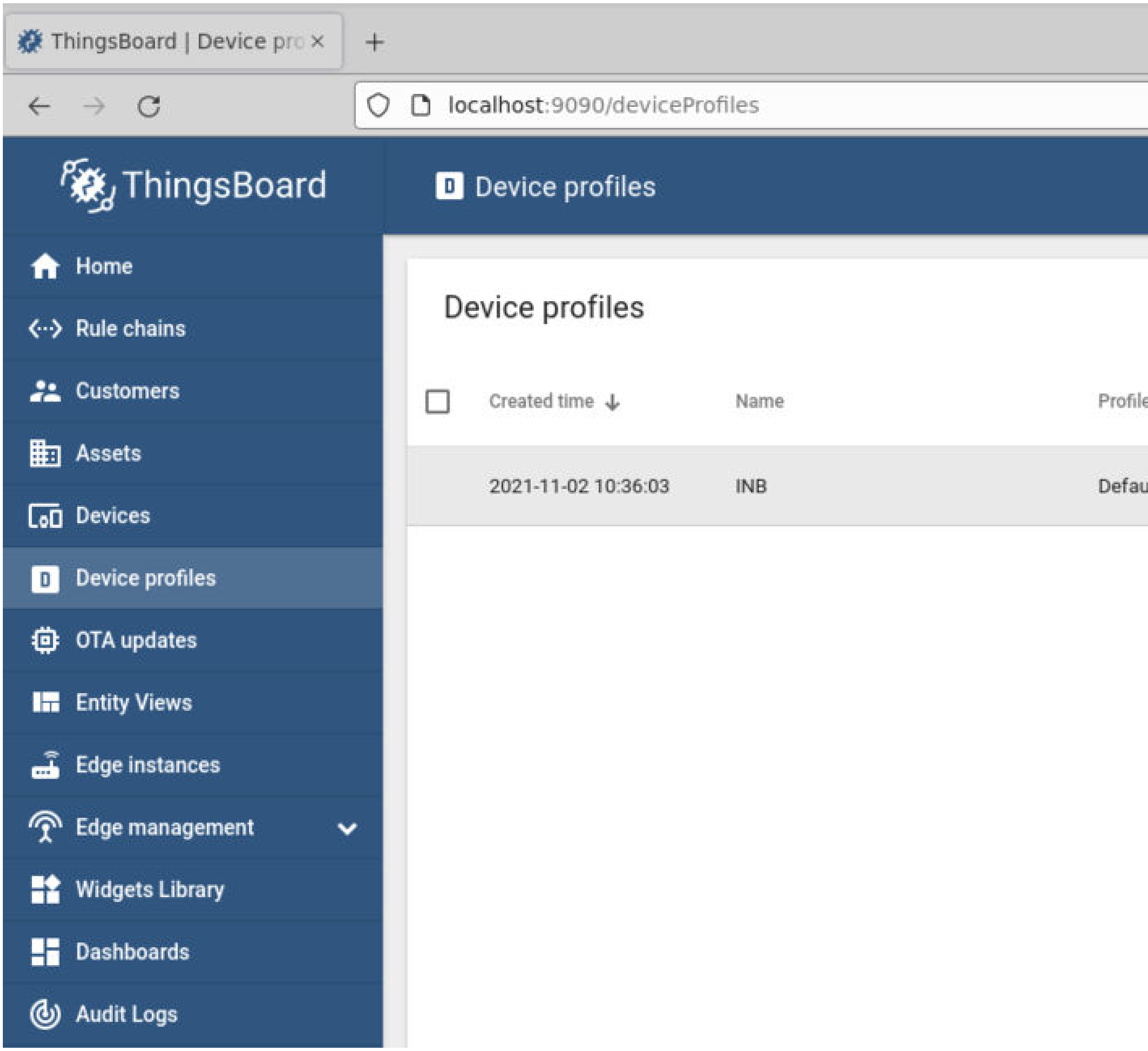
> **NOTE** Use the following credentials:
>
> - account: `tenant@thingsboard.org`
> - password: `tenant`
>
> If the Fleet Management Server Dashboard is not accessible on a system in the same network, check Troubleshooting for Robot Orchestration Tutorials, “Fleet Management Server Dashboard over LAN Issues”.

The following home page is loaded. **Device Profiles** and **Devices** are loaded with pre-configured data from Intel.



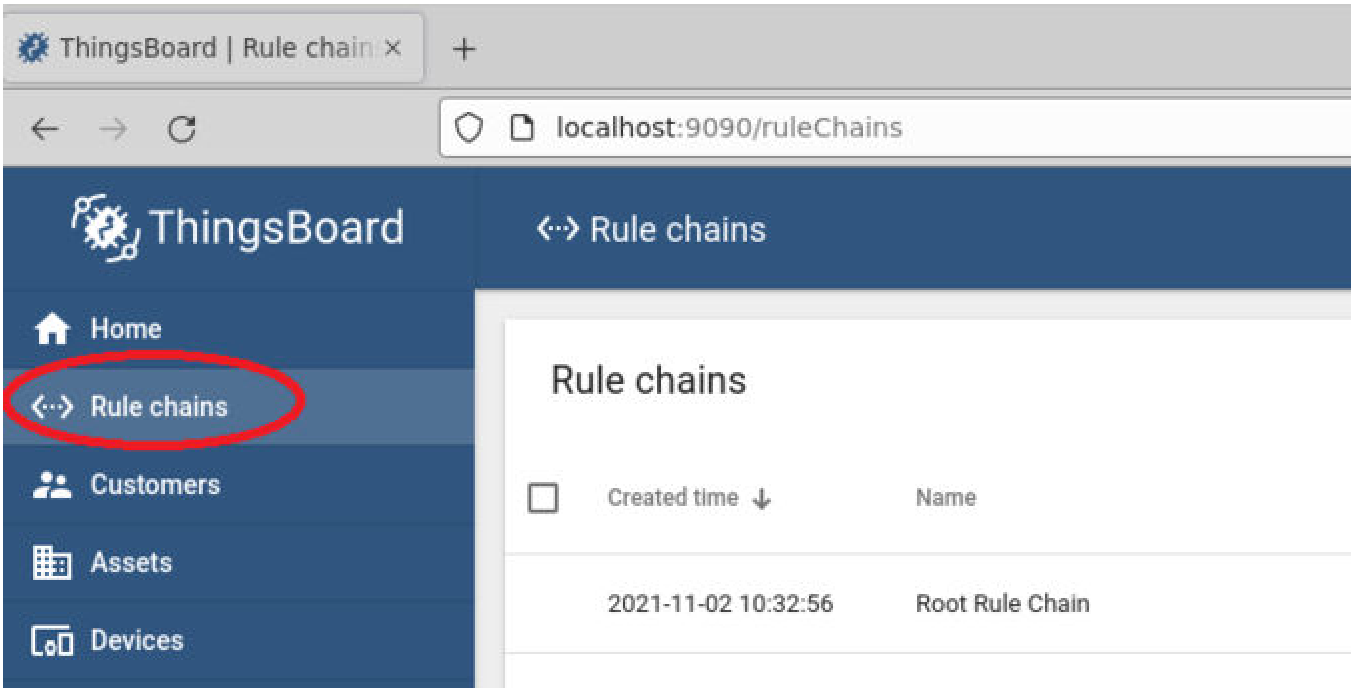
2. Check the pre-configured device profile from Intel named: INB.

- The Root Rule Chain is associated with the pre-defined Device Profile "INB."
- Definitions in this Root Rule Chain determine how incoming and outgoing messages are processed for all devices registered in this profile.



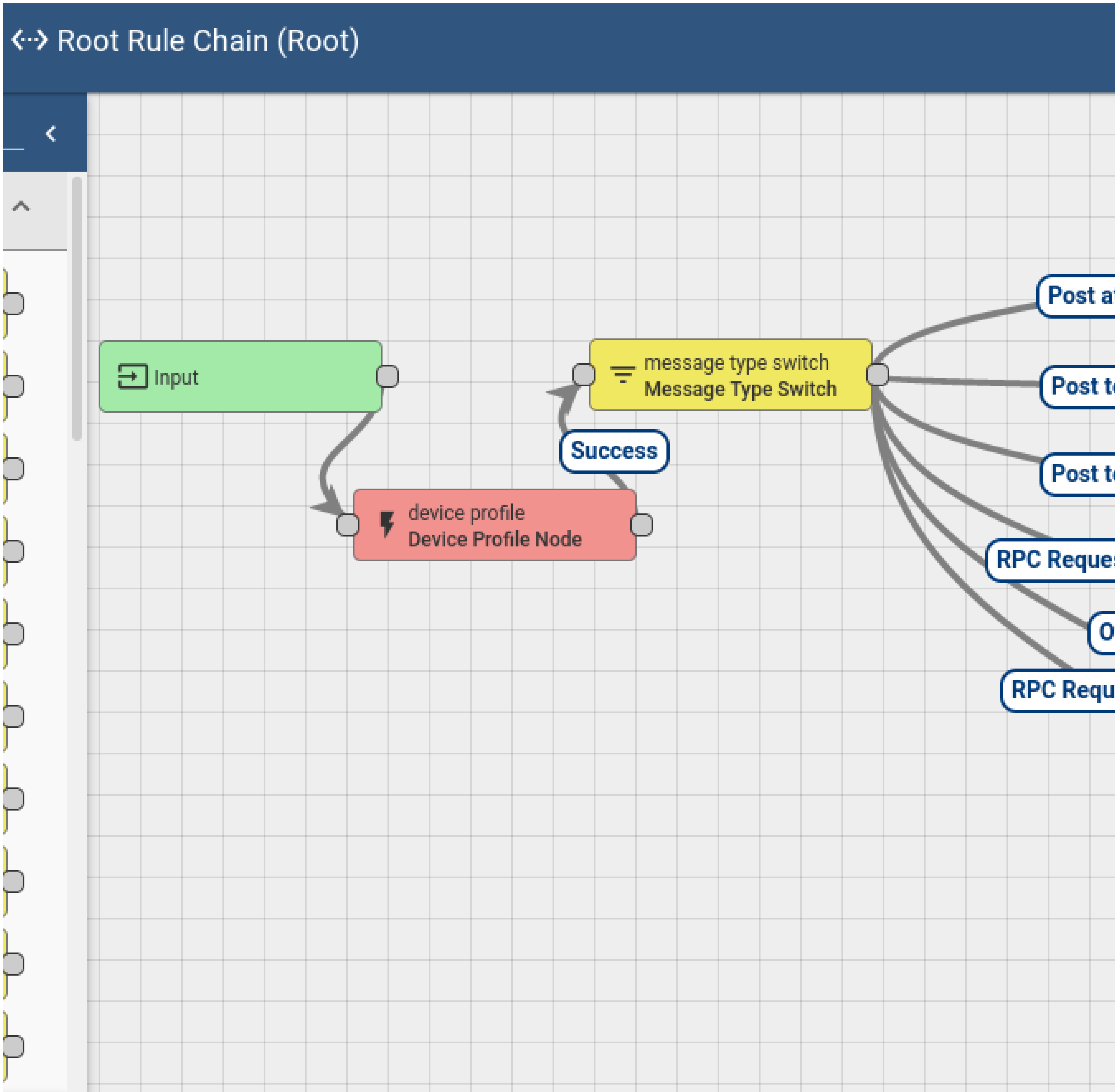
3. Check the pre-configured Root Rule Chain from ThingsBoard*.

a. Open the Rule:



b. Intel added the following to the Rule:

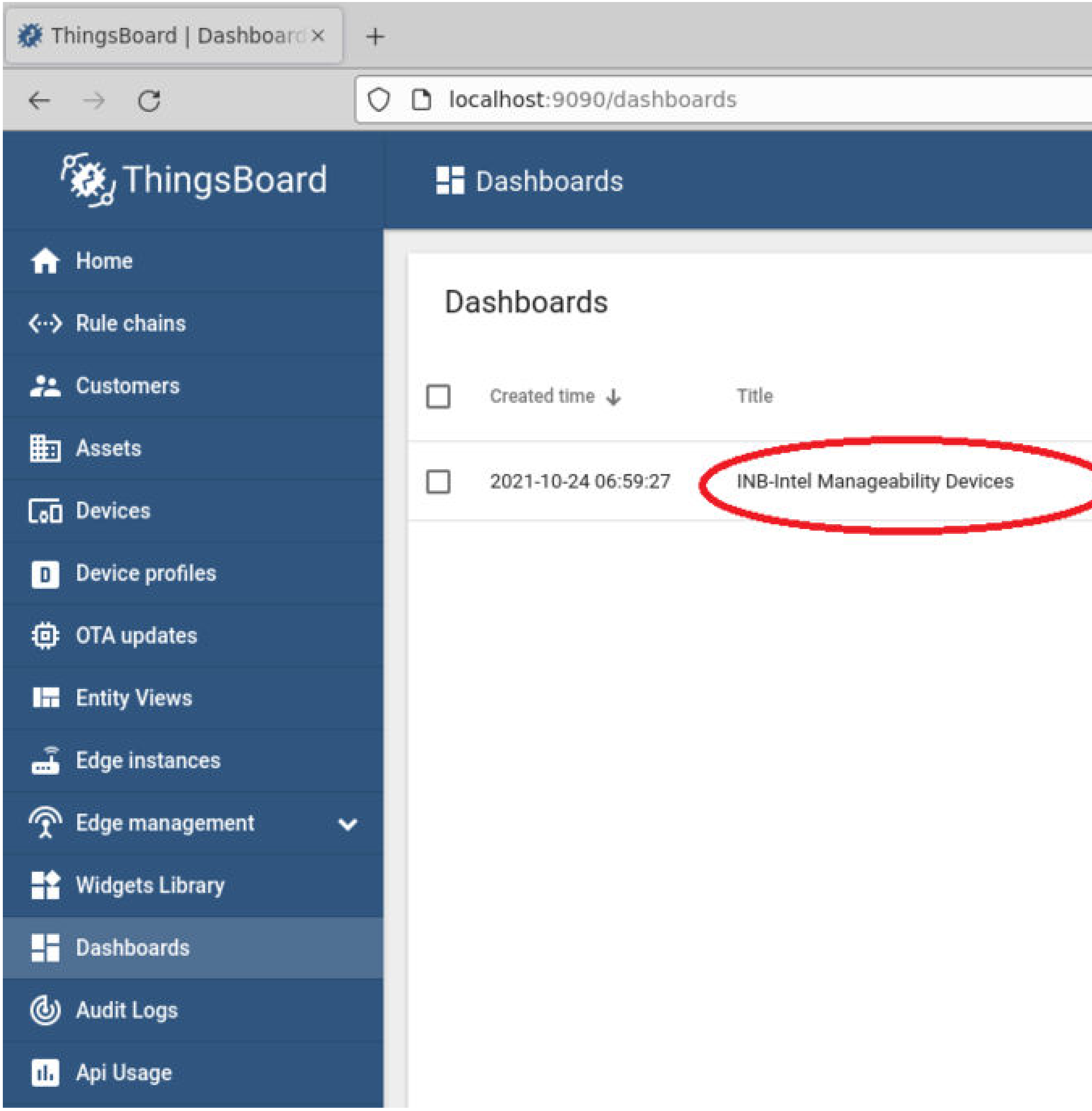
- The second PostTelemetry path (in order to not interfere with the original ThingsBoard* processing)
- The three circled nodes (in order to send the robot to the charging station when its battery is less than 40%)
  - The BatteryStatusCheck node validates the battery level: when the battery level is less than 40%, it is set to true to form the VDA5050 message.



NOTE To add new clients to the fleet management server, see Troubleshooting for Robot Orchestration Tutorials, "Add New Clients to the Fleet Management Server".

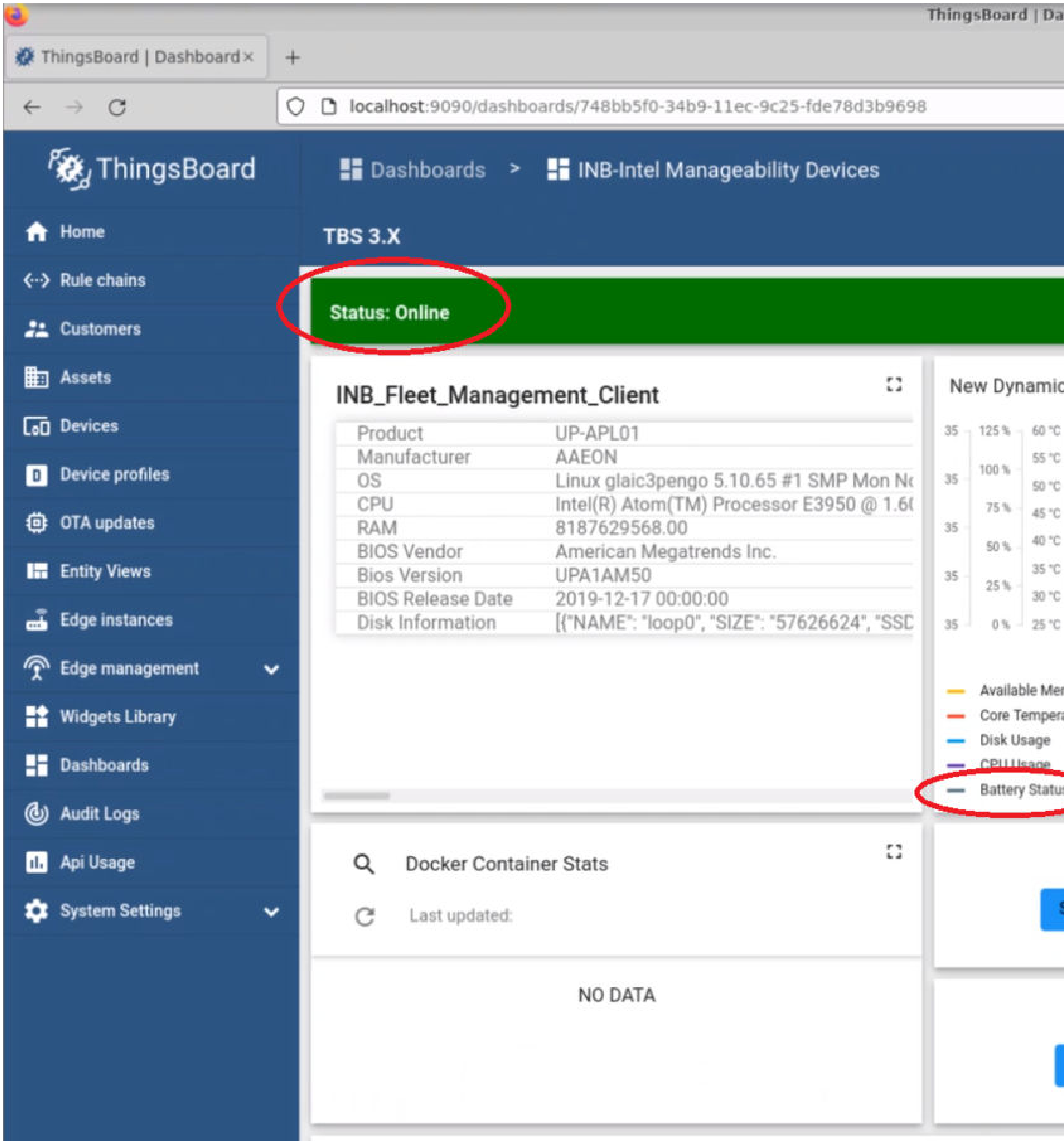
4. Check out the Dashboard tailored for Intel® In-Band Manageability.

a. Open the Dashboard:



b. The Dashboard shows Device basic information and telemetry data, for example:

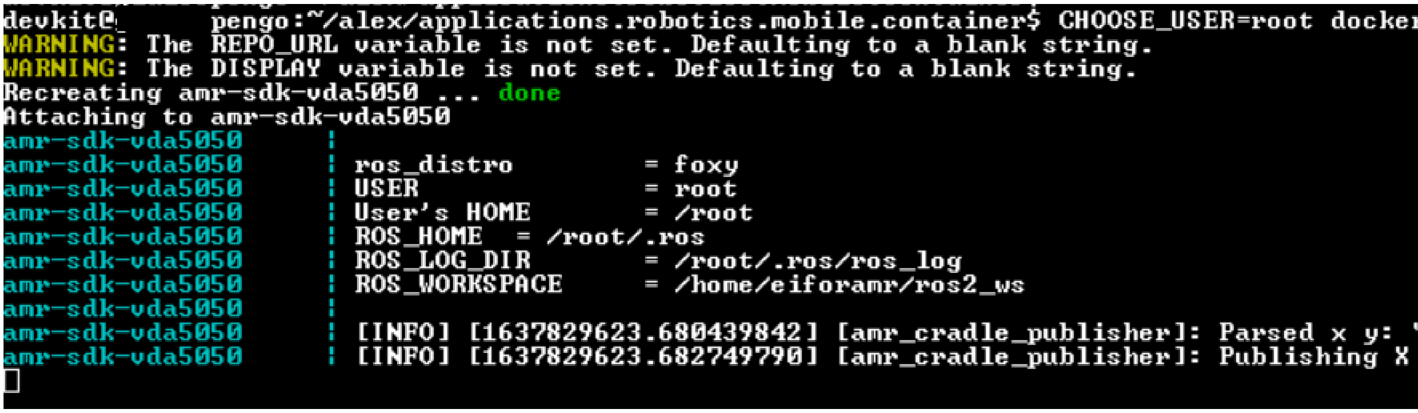
- The INB_Fleet_Management_Client device is currently online.
- The battery status is numeric and presented as an average value.
- The battery level is 35 (hover over the grey line representing battery value by time to see this).



NOTE Prerequisite: The battery reading is facilitated with a Python* psutil library call in Intel® In-Band Manageability. The function of psutil depends on the host where the basic fleet management client is deployed. Make sure the battery value is reflected in the psutil API call. If the battery is not reported as expected, see Troubleshooting for Robot Orchestration Tutorials, “Battery Status Not Available in Dashboard”.

5. Check the VDA event logs when the battery level goes under 40%.

Go to the separate VDA terminal. The logs are similar to this:



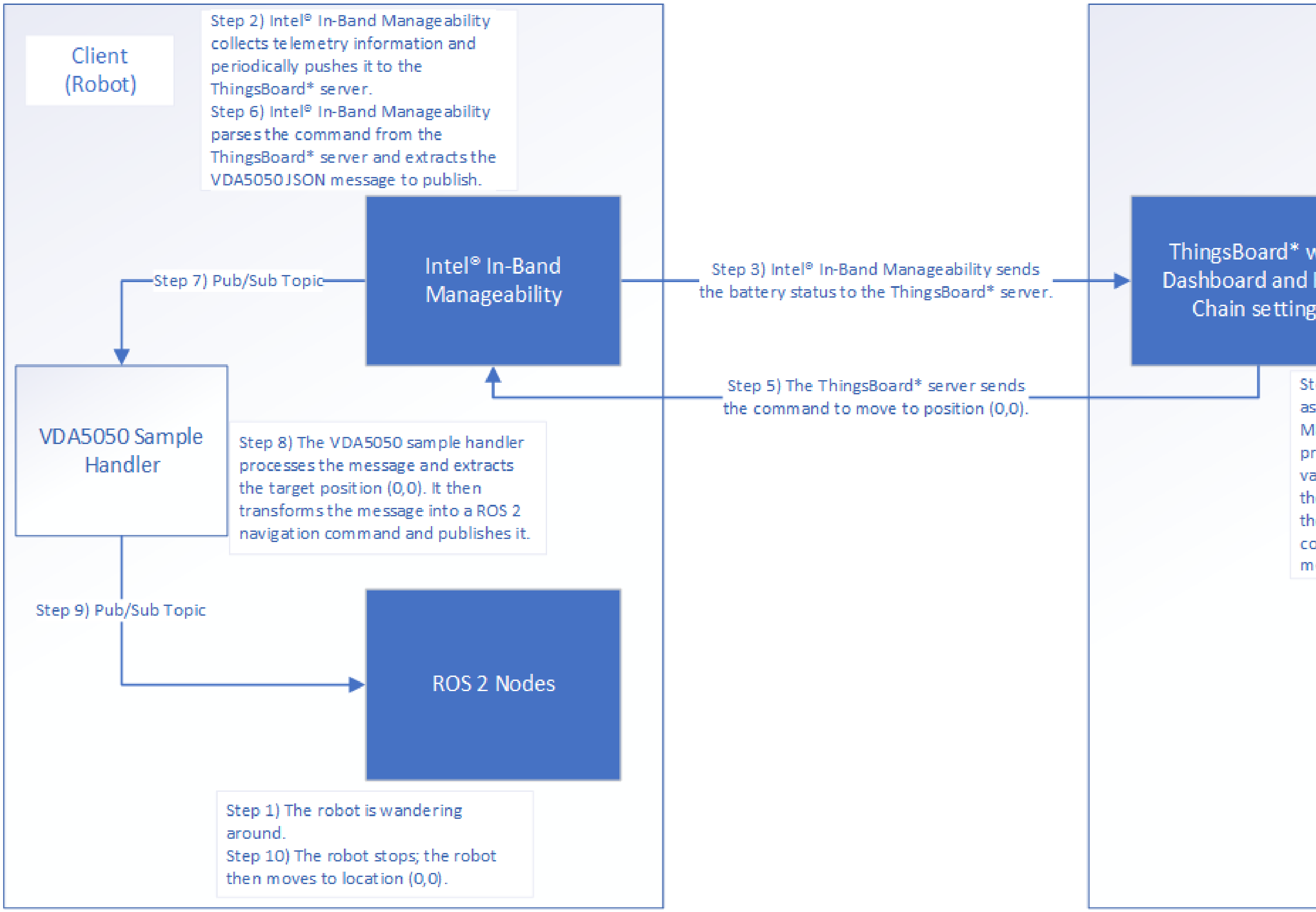
6. Check the Wandering application logs when the battery level goes under 40%.

The logs are similar to this:

```
[wandering_mapper]: GoToLocation
```

Collaboration Diagram

When a robot's battery level is less than 40%, basic fleet management tells the robot to move to the origin position. The following figure depicts the steps.



Remote Inference End-to-End Use Case

This tutorial describes how to use the basic fleet management server to set object detection inference on EI for AMR remotely at the OpenVINO™ model server when its battery is lower than the 60% threshold. If the battery is equal to or greather than 60%, the inference is set to be done locally at EI for AMR.

You must do all of the sections in this tutorial in the order listed.

- Machine A is the server.
- Machine B is an EI for AMR target that sends data to the server and receives instructions (the robot).
- Machine A and Machine B need to be in the same network.

Prerequisites: The robot and the server are configured as instructed in their Get Started Guides:

- Configuring the client: Get Started Guide for Robots
- Configuring the server (Single-Node or Multi-Node): Get Started Guide for Robot Orchestration
- Basic Fleet Management (both ThingsBoard* server and Intel® In-Band Manageability are required in the remote inference use case): Get Started Guide for Robot Orchestration

Configure the OpenVINO™ Model Server

On Machine A

1. Go to the `AMR_containers` folder:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_server_containers
```

2. Set up the necessary environment variables. Add the environment variables in `/etc/environment`:

```
export DOCKER_BUILDKIT=1
export COMPOSE_DOCKER_CLI_BUILD=1
export DOCKER_HOSTNAME=$(hostname)
export DOCKER_USER_ID=$(id -u)
export DOCKER_GROUP_ID=$(id -g)
export DOCKER_USER=$(whoami)
export DISPLAY=localhost:0.0
```

3. Source the configuration file you have edited:

```
source /etc/environment
```

4. Generate the keys used for server-client remote inference:

```
cd 01_docker_sdk_env/artifacts/02_edge_server/edge_server_fleet_management/ovms/
chmod +x generate_ovms_certs.sh
sudo ./generate_ovms_certs.sh <host-name-of-server>
```

> **NOTE** If Single-Node orchestration is deployed, the hostname of the Single-Node server shall be used to generate certificates. If Multi-Node orchestration is deployed, the hostname of the control plane shall be used to generate certificates.

5. Copy required client keys onto the client machine:

```
# From server folder `keys/` transfer (scp) server.pem, client.pem and client.key to client:
cd keys/
scp -r client.key client.pem server.pem <client_user>@<client_hostname>:~/
```

On Machine B

1. Copy the keys to /etc/amr/ri-certs:

```
mkdir -p /etc/amr/ri-certs
sudo cp ~/{client.key,client.pem,server.pem} /etc/amr/ri-certs
```

Start the OpenVINO™ Model Server

On Machine A

If Intel® Smart Edge Open Multi-Node is deployed, there are two machines for orchestration. Machine A-1 represents the controller, and Machine A-2 represents the server node where the OpenVINO™ model server pod is deployed.

If Intel® Smart Edge Open Single-Node is deployed, Machine A-1 and A-2 are the same machine.

1. Run the following command to avoid an Intel® Smart Edge Open playbook known limitation:

```
sed -i "s/edge-server-ubuntu2004-ovms-tls/edge-server-ovms-tls/g"  AMR_server_containers/
01_docker_sdk_env/docker_orchestration/ansible-playbooks/02_edge_server/openvino_model_server/
ovms_playbook_install.yaml
```

2. Go to `<edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<release>`, and run on Machine A-1:

```
ansible-playbook AMR_server_containers/01_docker_sdk_env/docker_orchestration/ansible-playbooks/
02_edge_server/openvino_model_server/ovms_playbook_install.yaml
```

3. Verify that the services, pods, and deployment are running on Machine A-1:

```
$ kubectl get all --output=wide --namespace ovms-tls

   NAME                                          READY   STATUS    RESTARTS      AGE
IP             NODE           NOMINATED NODE   READINESS GATES
   pod/ovms-deployment-75c7dffdc5-cgxp9         1/1     Running   1 (50m ago)   50m
10.245.202.9   glaic3roscube   <none>           <none>

   NAME                                          TYPE       CLUSTER-IP       EXTERNAL-IP
PORT(S)                         AGE   SELECTOR
   service/ovms-service                          NodePort   10.110.177.249   <none>
3335:32762/TCP,2225:32763/TCP   50m   app.kubernetes.io/instance=ovms-tls-
abcxzy,app.kubernetes.io/name=ovms-tls

   NAME                                          READY   UP-TO-DATE   AVAILABLE   AGE
CONTAINERS   IMAGES                                 SELECTOR
   deployment.apps/ovms-deployment               1/1     1            1           50m   ovms-
tls     10.237.23.153:30003/intel/ovms-tls:2022.2   app.kubernetes.io/instance=ovms-tls-
abcxzy,app.kubernetes.io/name=ovms-tls

   NAME                                          DESIRED   CURRENT   READY   AGE   CONTAINERS
IMAGES                                       SELECTOR
   replicaset.apps/ovms-deployment-75c7dffdc5   1         1         1       50m   ovms-tls
10.237.23.153:30003/intel/ovms-tls:2022.2   app.kubernetes.io/instance=ovms-tls-
abcxzy,app.kubernetes.io/name=ovms-tls,pod-template-hash=75c7dffdc5
```

NOTE CLUSTER-IP is a virtual IP that is allocated by Kubernetes* to a service. It is the Kubernetes* internal IP. Two different pods can communicate using this IP.

4. Verify that the Docker* container is running on Machine A-2:

```
docker ps | grep ovms-tls
6b64514a4b9a   c9e7db04fe06                       "/usr/bin/dumb-init …"   10 minutes ago   Up
10 minutes             k8s_ovms-tls_ovms-deployment-5856948447-t78tz_ovms-tls_3cd84b35-
c604-4948-9228-e381fd0714fa_1
ceeaa7673bcd   k8s.gcr.io/pause:3.5               "/pause"                 10 minutes ago   Up
10 minutes             k8s_POD_ovms-deployment-5856948447-t78tz_ovms-tls_3cd84b35-c604-4948-9228-
e381fd0714fa_1
```

If you encounter errors, see Troubleshooting for Robot Orchestration Tutorials.

Configure Object Detection with Remote Inference

On Machine B

1. Go to the `AMR_containers` folder:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_containers
```

2. Set up the environment:

```
export DOCKER_BUILDKIT=1
export COMPOSE_DOCKER_CLI_BUILD=1
export DOCKER_HOSTNAME=$(hostname)
export DOCKER_USER_ID=$(id -u)
export DOCKER_GROUP_ID=$(id -g)
export DOCKER_USER=$(whoami)
```

3. Make additional changes for remote inference:

a. Put the <host-name-of-server> used in the key generation step above inside the config file:

```
vim 05_tutorials/launch/remote_inference.launch.py
```

b. Update remote_hostname:

- For Intel® Smart Edge Open Multi-Node, use the control plane hostname.
- For Intel® Smart Edge Open Single-Node, use ThingsBoard* server pod's hostname.

```
<remote_hostname>: 32762
```

Start Object Detection with Remote Inference

On Machine B

1. When running the client on a robot:

```
CHOOSE_USER=root docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
aaeon_wandering_remote_inference.yml up object-detection realsense aaeon-amr-interface ros-base-
camera-tf collab-slam fastmapping nav2 wandering object-detection vda5050-ros2-bridge
```

If a battery bridge is installed, run:

```
CHOOSE_USER=root docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
aaeon_wandering_remote_inference.yml up battery-bridge object-detection realsense aaeon-amr-
interface ros-base-camera-tf collab-slam fastmapping nav2 wandering object-detection vda5050-
ros2-bridge
```

2. When running the client on a laptop or PC with a Intel® RealSense™ camera connected:

Before opening the Docker* images required to run object detection inference, make sure to set up your ROS_DOMAIN_ID in the common.yml file. See: Troubleshooting for Robot Tutorials, "Use ROS_DOMAIN_ID to Avoid Interference in ROS Messages".

```
CHOOSE_USER=root docker-compose -f 01_docker_sdk_env/docker_compose/05_tutorials/
aaeon_wandering_remote_inference.yml up battery-bridge realsense ros-base-camera-tf object-
detection vda5050-ros2-bridge
```

Because this use case is not executed on a robot with a real battery, you can switch between local and remote inference by manually setting the battery percentage values. In order to be able to do this, you need to have installed the battery kernel module. See: Basic Fleet Management Use Case, "Install the battery-bridge kernel module" for details.

After the battery-bridge kernel module is installed, you can adjust the battery percentage to different values.

Open a new terminal window, and update the battery percentage to a value equal to or higher than 60%:

```
echo 'capacity0 = 60' |  sudo tee /dev/battery_bridge
```

After setting the battery percentage to 60, the inference in the object-detection terminal should continue to run locally.

Set the battery percentage to a value below 60:

```
echo 'capacity0 = 59' |  sudo tee /dev/battery_bridge
```

After setting the battery percentage to 59, the object-detection should switch to run remote inference.

3. Example logs when Local Inference is performed:

```
[object_detection_node-3]

[object_detection_node-3] [ INFO ] <LocalInference> Done frame: 5516 . Processed in: 0.279625 ms

[object_detection_node-3]
```

```
[object_detection_node-3] [ INFO ] <LocalInference> Label tv

[object_detection_node-3]

[object_detection_node-3] [ INFO ] <LocalInference> Done frame: 5517 . Processed in: 0.240508 ms

[object_detection_node-3]

[object_detection_node-3] [ INFO ] <LocalInference> Label tv
```

4. Example logs when Remote Inference is performed:

```
[object_detection_node-3] [INFO] [1643382428.696445729] [object_detection]:
switchToRemoteInfCallback

[object_detection_node-3] [INFO] [1643382428.720869717] [object_detection]: <RemoteInference>
Sending Image

[object_detection_node-3] [INFO] [1643382428.854300655] [object_detection]: <RemoteInference>
Sending Image

[remote_inference-4] [INFO] [1643382428.863697253] [remote_inference]: <RemoteInference>
Receiving video frame

[object_detection_node-3] [INFO] [1643382428.882912426] [object_detection]: <RemoteInference>
Sending Image

[remote_inference-4] [INFO] [1643382428.895332223] [remote_inference]: <RemoteInference>
Processing and inference took 31.16

[object_detection_node-3] [INFO] [1643382428.896419726] [object_detection]: <RemoteInference>
Detected Objects Received

[object_detection_node-3] [INFO] [1643382428.896478543] [object_detection]: <RemoteInference>
Label : tv

[remote_inference-4] [INFO] [1643382428.897817637] [remote_inference]: <RemoteInference>
Receiving video frame

[object_detection_node-3] [INFO] [1643382428.921090305] [object_detection]: <RemoteInference>
Sending Image

[remote_inference-4] [INFO] [1643382428.922211172] [remote_inference]: <RemoteInference>
Processing and inference took 23.68
```

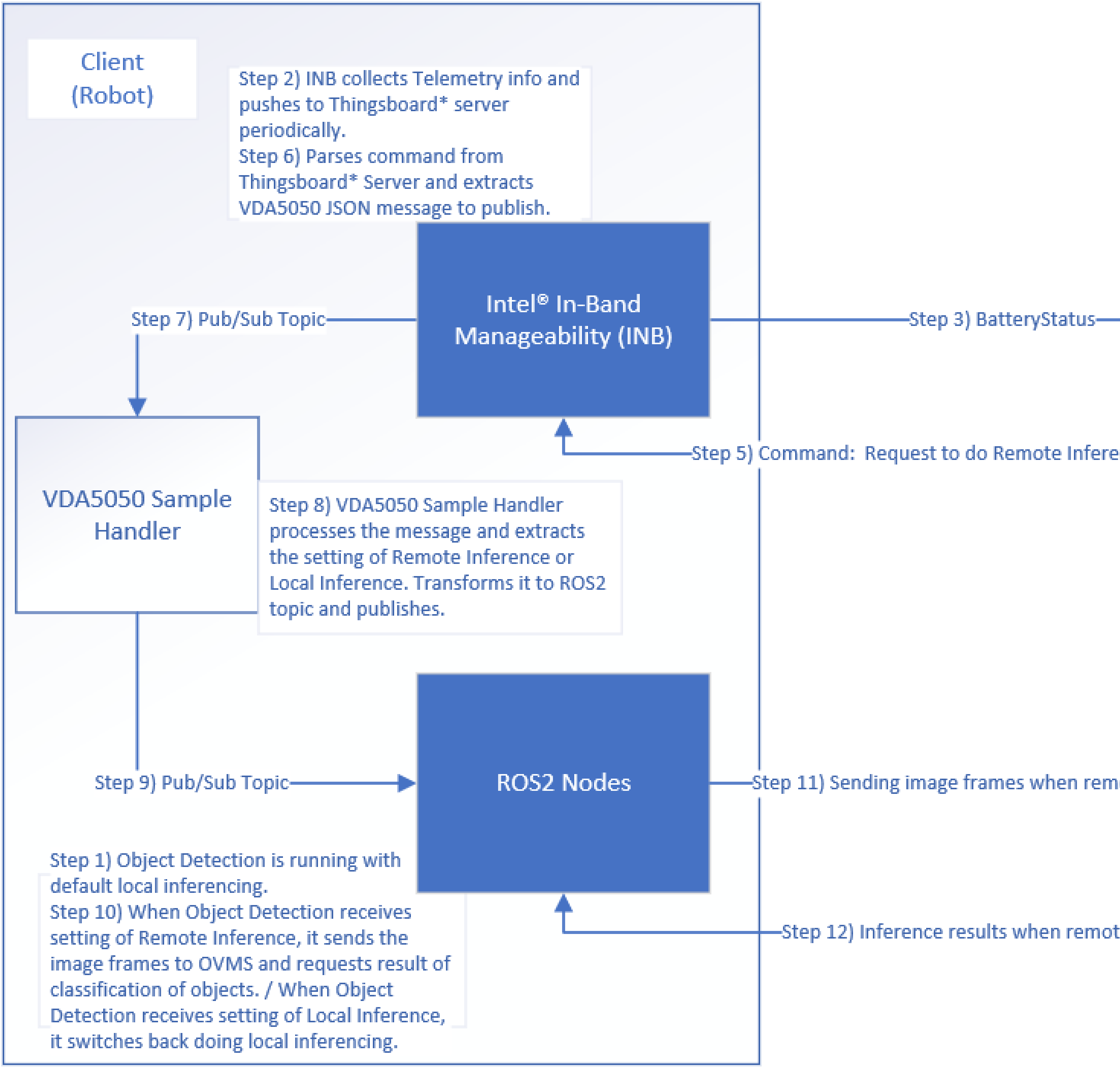
New Rules in the Basic Fleet Management Server

On Machine A

New rules in Root Chain are added to handle a battery level of less than 60% and equal to or greater than 60% scenarios.

Collaboration Diagram

When a robot’s battery level is less than 60%, basic fleet management tells the robots to do Remote Inference. When the battery level is back to equal or greater than 60%, basic fleet management tells the robots to do Local Inference. The following figure depicts the steps.

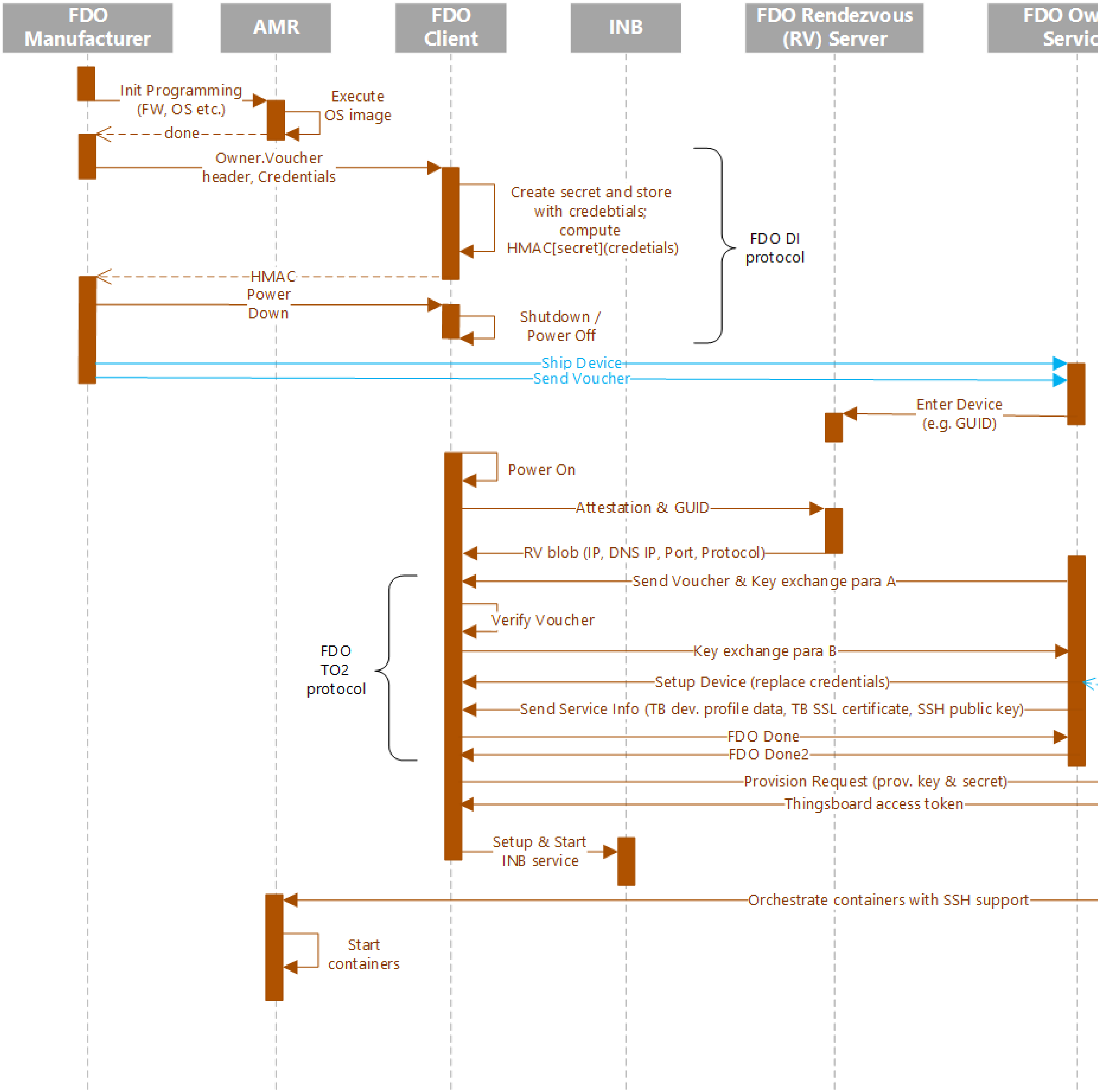


Device Onboarding End-to-End Use Case

This tutorial describes how to use the device onboarding system:

1. Onboard the Fast IDentity Online (FIDO) device.
2. Register the new device in ThingsBoard*.
3. Set up a secure TLS connection for communication.
4. Load specified applications (containers) to the EI for AMR device.

The following sequence chart is a simplified presentation of the onboarding flow:

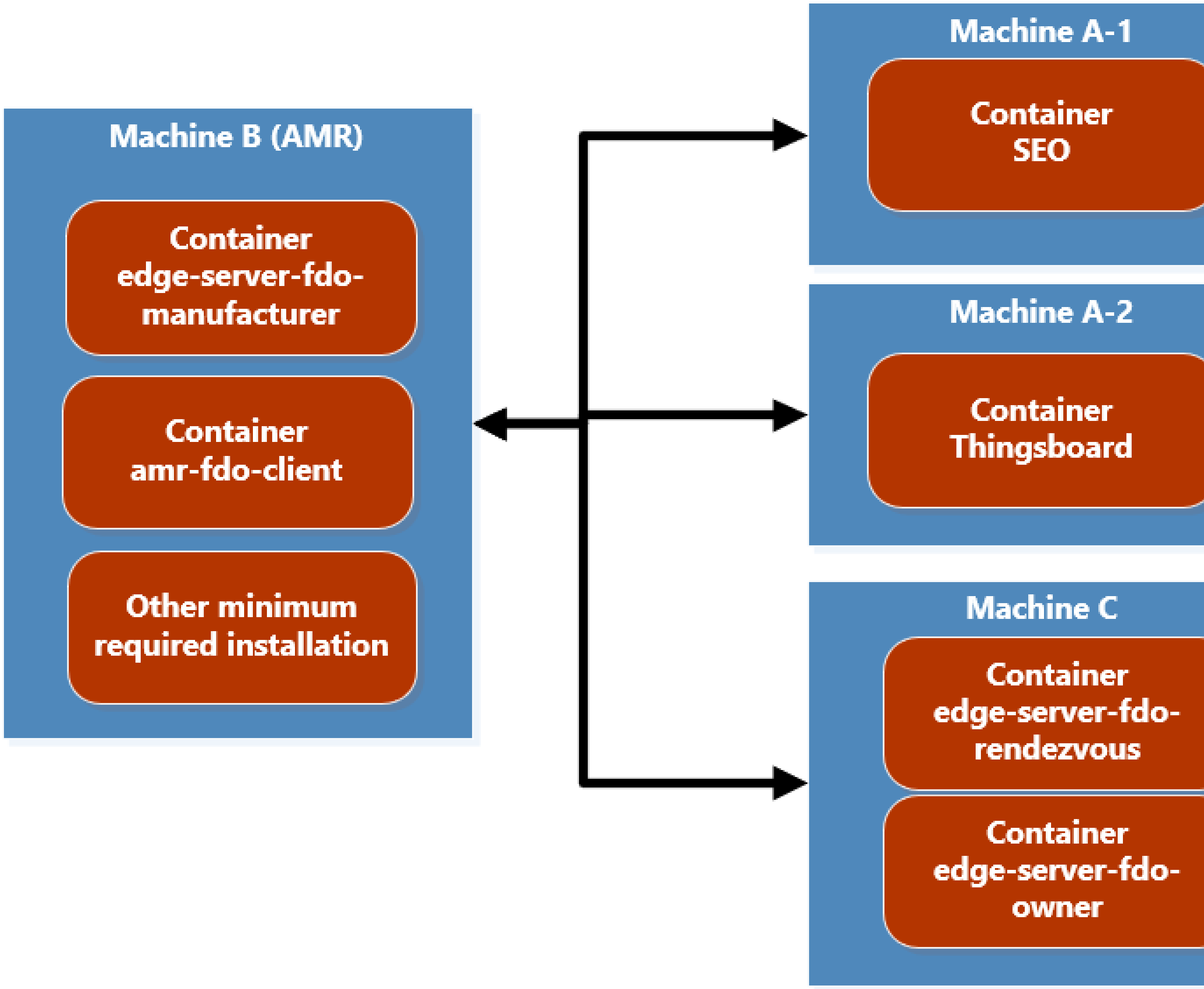


You must do all sections of this tutorial in order.

Prerequisites: The robot and the server are configured as instructed in the Get Started Guides.

- Configuring the server: Get Started Guide for Robot Orchestration.
- Configuring the robots: Get Started Guide for Robots.

For this use case, the following machines are used.



- Machine A-1 is the Intel® Smart Edge Open control plane which deploys ThingsBoard* to Machine A-2.

  NOTE The ThingsBoard* docker image is installed on a worker node on a different machine than the Intel® Smart Edge Open control plane. However, the ThingsBoard* GUI can be accessed using the Intel® Smart Edge Open control plane IP and mapped port.

- Machine B is the EI for AMR target that you want to onboard. **Machine B**:
  - executes amr-fdo-client in terminal **1**.
  - executes edge-server-fdo-manufacturer in terminal **2**.
  - uses terminal **3** for configuration and control.
- Machine C executes the rendezvous and owner servers:
  - edge-server-fdo-owner is on terminal **1**.
  - edge-server-fdo-rendezvous is on terminal **2**.

Minimum Install Requirements on the EI for AMR Device

1. **Machine B**: Download, and install the latest release.
    - a. Go to the Product Download page.
    - b. Select **Robot and Server Complete Kit**.
    - c. Click **Download**.
    - d. Copy the zip file to your target system.
    - e. Extract and install the software:

```
unzip edge_insights_for_amr.zip
cd edge_insights_for_amr
chmod 775 edgesoftware
export no_proxy="127.0.0.1/32,devtools.intel.com"
./edgesoftware download
./edgesoftware list
```

NOTE Get the ID for *Docker Community Edition CE* and for *Docker Compose*:

```
./edgesoftware update <ID_Docker Community Edition CE> <ID_Docker Compose>
./edgesoftware docker --pull amr-fleet-management:<docker_tag>
sudo groupadd docker
sudo usermod -aG docker $USER
newgrp docker
source /etc/environment
```

2. All images in the FDO pipeline are self-contained and require minimal configuration. Configuration settings are all handled by external environment files. But some environment files need to be generated by running the `fdo_keys_gen.sh` script:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_server_containers/01_docker_sdk_env/artifacts/02_edge_server/edge_server_fdo/
chmod +x fdo_keys_gen.sh
bash fdo_keys_gen.sh .
```

3. Copy the generated certificates to Machine C:

```
scp -r creds/ machine_c_user@machine_c_ip:/<path_to_edge_insights_for_amr>/AMR_server_containers/
01_docker_sdk_env/artifacts/02_edge_server/edge_server_fdo/
```

4. Make sure that password-less `ssh` access for `root` is set on all machines:

```
sudo nano /etc/ssh/sshd_config
```

    - a. Add the following line at the end of the file:

```
PermitRootLogin yes
```

    - b. After the `/etc/ssh/sshd_config` is updated, restart the `ssh` service:

```
sudo service ssh restart
sudo su
service ssh restart
ssh-keygen
exit
```

Build FDO Docker* Images

These steps have to be re-executed if a terminal is closed.

1. **Machine B, C- all terminals:**

```
export DISPLAY=0:0
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_containers/
```

2. **Machine B- all terminals:**

```
export no_proxy=<no_proxy>,ip_from_machine_C,ip_from_machine_B,localhost
sudo su
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
docker tag amr-fleet-management:<docker_tag> amr-fleet-management:latest
```

3. **Machine B, C- all terminals:** Prepare the environment setup:

```
source 01_docker_sdk_env/docker_compose/05_tutorials/config/docker_compose.source
```

> **NOTE** Set up the environment in each new terminal on which you want to run `docker-compose` commands.

4. **Machine C- terminal 1:** Get the DNS:

```
sudo cat /run/systemd/resolve/resolv.conf
```

5. **Machine B- terminal 1**: Build the FDO client:

   Before building the FDO Client image, there are a variety of configuration flags that can be adjusted.

   a. Open 01_docker_sdk_env/artifacts/02_edge_server/edge_server_fdo/manufacturer/service.yml:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_server_containers
nano 01_docker_sdk_env/artifacts/02_edge_server/edge_server_fdo/manufacturer/service.yml
```

   b. Add the following lines:

```
# Modify the values shown below in bold in the above file with respective DNS and IP address of
Rendezvous server
rv-instruction:
   dns: dns_from_step_4
   ip: ip_from_machine_C
```

   c. Build the image:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_containers
docker-compose -f ./01_docker_sdk_env/docker_compose/01_amr/amr-sdk.all.yml build fdo-client
```

6. **Machine B- terminal 2**: Build the manufacturer server image:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_containers
docker-compose -f ./01_docker_sdk_env/docker_compose/02_edge_server/edge-server.all.yml build
fdo-manufacturer
```

7. **Machine C- terminal 1**: Build the owner server image:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_containers
docker-compose -f ./01_docker_sdk_env/docker_compose/02_edge_server/edge-server.all.yml build
fdo-owner
```

8. **Machine C- terminal 2**: Build the rendezvous server image:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_containers
docker-compose -f ./01_docker_sdk_env/docker_compose/02_edge_server/edge-server.all.yml build
fdo-rendezvous
```

Set Up the Intel® Smart Edge Open Controller on Machine A-1

1. Make sure that the common name is the hostname of Machine A-1.
2. Install the Eclipse Mosquitto* broker and client for device onboarding and application over-the-air (AOTA) message queuing telemetry transport (MQTT) messages:

```
sudo apt-add-repository ppa:mosquitto-dev/mosquitto-ppa
sudo apt-get update
sudo apt-get install mosquitto
sudo apt-get install mosquitto-clients
sudo apt clean
```

3. Generate the certificates for the Mosquitto secure sockets layer (SSL) and the `server.key` for the Mosquitto SSL:

```
cd  /etc/mosquitto/certs
openssl genrsa -des3 -out ca.key 2048
openssl req -new -x509 -days 1826 -key ca.key -out ca.crt
openssl genrsa -out server.key 2048
openssl req -new -out server.csr -key server.key
```

```
openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out server.crt -days
360
```

NOTE Use the machine hostname of Machine A as the common name.

4. Update `/etc/mosquitto/mosquitto.conf`:

```
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d

listener 18883
allow_anonymous true

cafile /etc/mosquitto/certs/ca.crt
certfile /etc/mosquitto/certs/server.crt
keyfile /etc/mosquitto/certs/server.key
```

5. Go to `<edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<release>`, and update the mqtt_onboard_aota.py file with the control plane hostname:

```
cd 01_docker_sdk_env/docker_orchestration/ansible-playbooks/02_edge_server/smart_edge_open/
chmod +x *.sh
nano mqtt_onboard_aota.py
```

```
DEFAULT_MQTT_HOST = "<Machine_A_HOSTNAME>
```

6. Start the MQTT onboard AOTA script:

```
ufw allow 18883
python3 mqtt_onboard_aota.py &
```

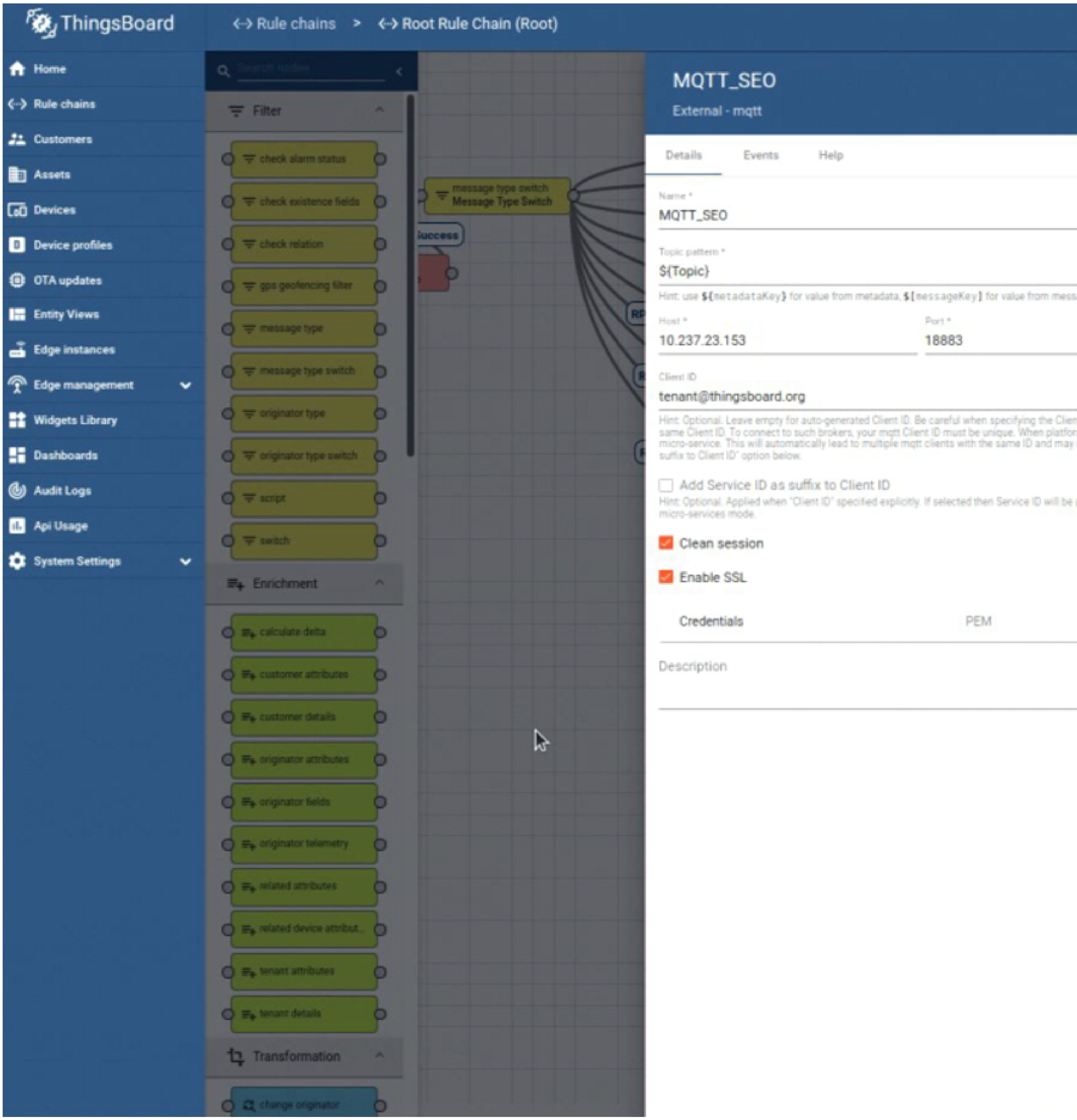
For errors, go to Troubleshooting.

Set Up ThingsBoard* on Machine A-2

1. Open a browser, use the controller IP, and open <IP Address>:32764. Use the following credentials:

- account: `tenant@thingsboard.org`
- password: `tenant`

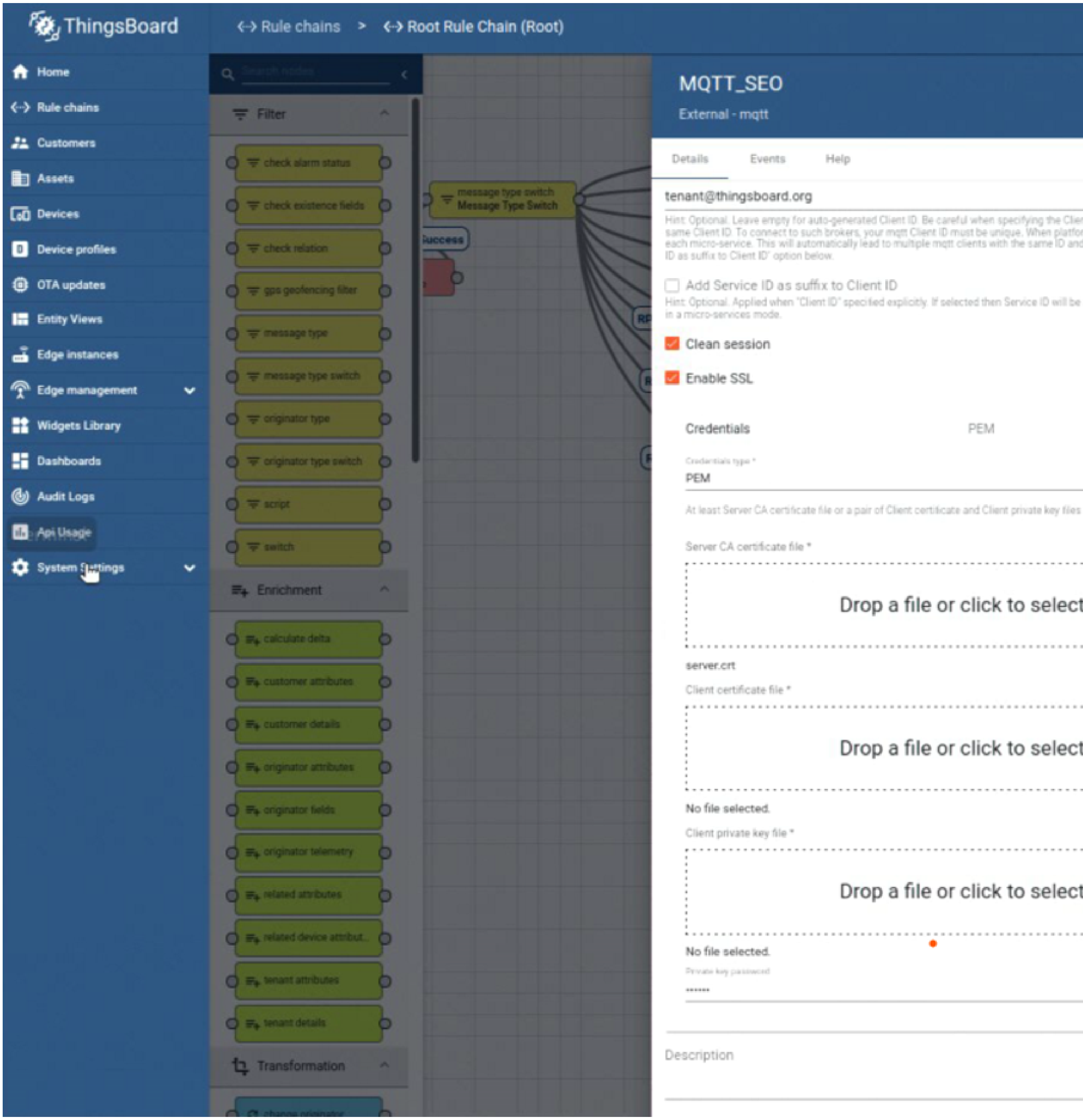
2. Go to the **Rule Chain** page, and select `MQTT_SEO`.



3. Assign the Machine A-1 IP to the variable `Host*`.

4. Select the `Enable SSL` option.

5. Assign `PEM` to the variable `Credentials`.
6. Upload the `/etc/mosquitto/certs/server.crt` certificate that was generated above, and apply the changes.



Initialize FDO

1. **Machine B- terminal 3:** FDO has to transfer some information (e.g. ThingsBoard* device provision token or TLS certificate) to the fleet-management Intel® In-Band Manageability container. For this purpose the host folder ‘’/etc/tc’’ is used.

```
sudo mkdir /etc/tc
```

2. **Machine B- terminal 3:** Adjust the Python script 01_docker_sdk_env/artifacts/02_edge_server/edge_server_fdo/scripts/register_with_tb.py to your setup:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_server_containers/
nano 01_docker_sdk_env/artifacts/02_edge_server/edge_server_fdo/scripts/register_with_tb.py
```

Update with the following values:

a. Assign the Machine A IP to the variable `tb_address`
b. Assign the ThingsBoard* HTTP port to the variable `tb_port` (for example, 9000).

NOTE For Intel® Smart Edge Open setup the tb_port is 32764

c. Assign the values for `device_key` and `device_secret` with the values obtained from the ThingsBoard* web interface. Go to Thingsboard > Device Profiles > Device Profiles details > Device Provisioning.

In preconfigured data, the following are set in ThingsBoard*:

```
device_key = "9oq7uxtdsgt4yjyqdekg"
device_secret = "6z3j3osphpr8ck1b9ocp"
```

3. **Machine B- terminal 3:** Adjust the Python script `01_docker_sdk_env/artifacts/02_edge_server/edge_server_fdo/scripts/register_with_tbtc.py` for your setup. Use a block copy to follow the script file formatting.

```
nano 01_docker_sdk_env/artifacts/02_edge_server/edge_server_fdo/scripts/register_with_tbtc.py
```

Update with the following values:

a. Assign the Machine C IP to the variable `IP`.
b. Assign the Machine A-1 IP to the variable `tb_address`.
c. Assign the ThingsBoard* HTTP port to the variable `tb_port` (for example, 9000).

NOTE For Intel® Smart Edge Open setup, `tb_port` is 32764.

d. Assign the ThingsBoard* MQTT TLS port to the variable `tb_mqtt_port` (for example, 8883).

NOTE For Intel® Smart Edge Open setup, `tb_mqtt_port` is 32767.

e. **Machine A-1:** Get the SSH public key, and assign it to the variable `ssh_public_key`. The public key is generated by the `ssh-keygen` tool.
f. **Machine A-2:** Get the ThingsBoard* TLS certificate, and assign it to the variable `tb_pem` (refer to certificate generation in the Get Started Guide for Robot Orchestration. The certificate is used in Intel® In-Band Manageability and ThingsBoard* transport layer security [TLS] communication.).

Example:

```
...........................
s.connect(('10.255.255.255', 1))
         IP = s.getsockname()[0]
```

```
      except Exception:
         IP = 'ip_from_machine_C'
      finally:
         s.close()
      return IP

# Thingsboard IP-Address. Default is localhost, update if multisystem is desired
tb_address = http://<machine_A_IP>   # TB Server
# Thingsboard Docker Image http port
tb_port =  9090  # for SEO use 32764
# Thingsboard Docker Image mqtt port (typical 1883-nonSSL or 8883 SSL); req. by TurtleCreek
tb_mqtt_port = 8883  # for SEO use 32767
...........................
# Important: Don't use this certificate, replace it by your own!!
tb_pem = "-----BEGIN CERTIFICATE-----\n"+\

   Add the output of "cat /etc/thingsboard/conf/server.pub.pem " command from MACHINE A-2

"-----END CERTIFICATE-----\n"
...........................

# This key-value is an example and there is no matching private key
# Important: create your own SSH key and copy the public key here in the string.

ssh_public_key = "the output of "cat ~/.ssh/id_rsa.pub" command from MACHINE A-1"
```

4. **Machine B- terminal 3:** Edit `01_docker_sdk_env/artifacts/02_edge_server/edge_server_fdo/scripts/multi_machine_config.sh`, and set the following variables:

```
nano 01_docker_sdk_env/artifacts/02_edge_server/edge_server_fdo/scripts/multi_machine_config.sh
```

Update with the following values:

a. Assign the value from `01_docker_sdk_env/artifacts/02_edge_server/edge_server_fdo/creds/manufacturer/service.env` to the variable `mfg_api_passwd`.

```
cat 01_docker_sdk_env/artifacts/02_edge_server/edge_server_fdo/creds/manufacturer/service.env
```

b. Assign the value from `01_docker_sdk_env/artifacts/02_edge_server/edge_server_fdo/creds/owner/service.env` to the variable `default_onr_api_passwd`.

```
cat 01_docker_sdk_env/artifacts/02_edge_server/edge_server_fdo/creds/owner/service.env
```

c. Assign the Machine C DNS to the variables `rv-dns` and `owner-dns`.

d. Assign the Machine C IP to the variables `rv-ip` and `owner-ip`.

e. Replace the http://localhost:8042 in the second curl command with http://MACHINE_C_IP:8042.

Example (without the curly brackets):

```
mfg_api_passwd={manufacturer_api_password_from_service.env}
onr_api_passwd={owner_api_password_from_service.env}
.......................................................
# Updating RVInfo blob in Manufacturer
# Replace localhost, {rv-dns} and {rv-ip} references with respective DNS and IP address of the
host machine
curl -D - --digest -u "${api_user}":"${mfg_api_passwd}" --location --request POST 'http://
localhost:8039/api/v1/rvinfo' \
--header 'Content-Type: text/plain' \
--data-raw '[[[5,"dns"],[3,8040],[12,1],[2,"ip_from_machine_C"],[4,8040]]]'

# Updating T02RVBlob in Owner
# Replace localhost, {owner-ip} and {owner-dns} references with respective DNS and IP address of
the host machine
```

```
curl -D - --digest -u "${api_user}":"${onr_api_passwd}" --location --request POST 'http://
<ip_from_machine_c>:8042/api/v1/owner/redirect' \
--header 'Content-Type: text/plain' \
--data-raw '[["ip_from_machine_C","dns",8042,3]]'
```

5. **Machine B- terminal 3:** Edit `01_docker_sdk_env/artifacts/02_edge_server/edge_server_fdo/scripts/extend_upload.sh`, and set the following variables:

```
nano  01_docker_sdk_env/artifacts/02_edge_server/edge_server_fdo/scripts/extend_upload.sh
```

Update with the following values:

a. Assign the value from `01_docker_sdk_env/artifacts/02_edge_server/edge_server_fdo/creds/manufacturer/service.env` to the variable `default_mfg_api_passwd`.

```
cat 01_docker_sdk_env/artifacts/02_edge_server/edge_server_fdo/creds/manufacturer/service.env
```

b. Assign the value from `01_docker_sdk_env/artifacts/02_edge_server/edge_server_fdo/creds/owner/service.env` to the variable `owner_api_password_from_machine`.

```
cat 01_docker_sdk_env/artifacts/02_edge_server/edge_server_fdo/creds/owner/service.env
```

c. Assign the Machine B IP to the variable `default_mfg_ip`.
d. Assign the Machine C IP to the variable `default_onr_ip`.

Example:

```
default_mfg_ip="<ip_from_machine_B>"
default_onr_ip="<ip_from_machine_C>"
...........................
default_mfg_api_passwd="<manufacturer_api_password_from_service.env>"
default_onr_api_passwd="<owner_api_password_from_service.env>"
```

6. **Machine B- terminal 3:** Edit `01_docker_sdk_env/artifacts/02_edge_server/edge_server_fdo/scripts/configure_serviceinfo.sh`, and set the following variables:

a. Assign the Machine C IP to the variable `OWNER_IP`.

Onboard

FDO is a new IoT standard that is built on Intel® Secure Device Onboard (Intel® SDO) specifications. It is the first step in onboarding a device. The FDO specification specifies four entities.

- Device: the EI for AMR device plus the FDO client (the FDO client supports the FDO protocol)
- Manufacturer Server: the entity that is responsible for the initial steps of the FDO protocol and loading credentials onto the device, and is also a part of the production flow of the MAR device
- Owner Server: the entity that sends all required data (for example, keys and certificates) to the device in the final protocol step TO2
- Rendezvous Server: the first contact point for the device after you switch the device on and configure it for network communication. The rendezvous server sends the device additional information, for example, how to contact the owner server entity.

All containers, including the client, follow this command structure:

```
docker-compose -f <.yml path used during build stage> up <fdo service name>
```

1. **Machine B- terminal 2:** Run the manufacturer server:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_server_containers/
docker-compose -f 01_docker_sdk_env/docker_compose/02_edge_server/edge-server.all.yml up fdo-
manufacturer
```

2. **Machine C- terminal 1:** Run the owner server:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_server_containers/
docker-compose -f 01_docker_sdk_env/docker_compose/02_edge_server/edge-server.all.yml up fdo-
owner
```

3. **Machine C- terminal 2:** In a new terminal window, run the rendezvous server:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_server_containers/
docker-compose -f 01_docker_sdk_env/docker_compose/02_edge_server/edge-server.all.yml up fdo-
rendezvous
```

4. **Machine B- terminal 1:** Run the client:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_server_containers/
sudo su
docker-compose -f 01_docker_sdk_env/docker_compose/01_amr/amr-sdk.all.yml up fdo-client
```

After running the FDO client for the first time, the device initialization is complete:

```
FDO Client log snippet:

amr-sdk-fdo-client     | 09:56:55:433 FDOProtDI: Received message type 13 : 1 bytes
amr-sdk-fdo-client     | 09:56:55:433 Writing to Normal.blob blob
amr-sdk-fdo-client     | 09:56:55:433 Hash write completed
amr-sdk-fdo-client     | 09:56:55:434 HMAC computed successfully!
amr-sdk-fdo-client     | 09:56:55:434 Writing to Secure.blob blob
amr-sdk-fdo-client     | 09:56:55:434 Generating platform IV of length: 12
amr-sdk-fdo-client     | 09:56:55:434 Generating platform AES Key of length: 16
amr-sdk-fdo-client     | 09:56:55:434 Device credentials successfully written!!
amr-sdk-fdo-client     | (Current) GUID after DI: <GUID>
amr-sdk-fdo-client     | 09:56:55:434 DIDone completed
amr-sdk-fdo-client     | 09:56:55:434
amr-sdk-fdo-client     | ------------------------------------ DI Successful
--------------------------------------
amr-sdk-fdo-client     | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
amr-sdk-fdo-client     | @FIDO Device Initialization Complete@
amr-sdk-fdo-client     | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
amr-sdk-fdo-client exited with code 0
```

NOTE When starting FDO containers, start the FDO client image last because the FDO client image immediately begins reaching out to the manufacturer server in order to complete device initialization (DI), and it only attempt this connection a few times before exiting. If the FDO client is successful in connecting to the manufacturer server, the manufacturer server assigns a GUID to the FDO client and generates an ownership voucher for use in the rest of the pipeline.

5. **Machine B- terminal 3:** Move into the script folder.

NOTE Run the FDO scripts on Machine B as `root`.

```
cd 01_docker_sdk_env/artifacts/02_edge_server/edge_server_fdo/scripts/
chmod +x *
sudo su
export no_proxy=<no_proxy>,ip_from_machine_C,ip_from_machine_B,localhost
```

6. **Machine B- terminal 3:**

```
./multi_machine_config.sh
```

Expected output:

```
HTTP/1.1 401
WWW-Authenticate: Digest realm="Authentication required", qop="auth",
nonce="1652260953609:a1f80c513623b4c7b87292c054d5d650", opaque="4F6AB1DF45A94C67D59892BC7DB6B6B4"
Content-Type: text/html;charset=utf-8
Content-Language: en
Content-Length: 673
Date: Wed, 11 May 2022 09:22:33 GMT

HTTP/1.1 200
Content-Length: 0
Date: Wed, 11 May 2022 09:22:33 GMT

HTTP/1.1 401
WWW-Authenticate: Digest realm="Authentication required", qop="auth",
nonce="1652260953705:0e2856e16da3eb830dca777a34f1f154", opaque="E11DE6169652A5495FC93933790D1A04"
Content-Type: text/html;charset=utf-8
Content-Language: en
Content-Length: 673
Date: Wed, 11 May 2022 09:22:33 GMT

HTTP/1.1 200
Content-Length: 0
Date: Wed, 11 May 2022 09:22:33 GMT
```

7. **Machine B- terminal 3:**

```
./configure_serviceinfo.sh
```

Expected output:

```
Upload Device execution script to Owner Server
HTTP/1.1 401
WWW-Authenticate: Digest realm="Authentication required", qop="auth",
nonce="1652941145981:e5cdb0c180cd069360cd159fdcadccde", opaque="BE4E73265635CC0D98F9430BABA64DBE"
Content-Type: text/html;charset=utf-8
Content-Language: en
Content-Length: 673
Date: Thu, 19 May 2022 06:19:05 GMT

HTTP/1.1 100

HTTP/1.1 200
Content-Length: 0
Date: Thu, 19 May 2022 06:19:05 GMT
```

8. **Machine B- terminal 3:**

```
./extend_upload.sh -s 1234abcd
```

Expected output:

```
Success in downloading SECP256R1 owner certificate to owner_cert_SECP256R1.txt
Success in downloading extended voucher for device with serial number 1234abcd
Success in uploading voucher to owner for device with serial number 1234abcd
GUID of the device is 7e1e0c59-6d87-4b40-b68d-e7fcc00a7e37
Success in triggering TO0 for 1234abcd with GUID 7e1e0c59-6d87-4b40-b68d-e7fcc00a7e37 with
response code: 200
xxxx@machineA: 01_docker_sdk_env/artifacts/02_edge_server/edge_server_fdo/scripts$
```

9. **Machine C- terminal 1:** In the `edge-server-fdo-owner` logs, verify that TO0 finished.

```
edge-server-fdo-owner | 06:49:50.463 [INFO ] TO0 completed for GUID: ...
```

NOTE This task can take more than three minutes.

10. **Machine B- terminal 1:**

```
docker-compose -f 01_docker_sdk_env/docker_compose/01_amr/amr-sdk.all.yml up fdo-client
```

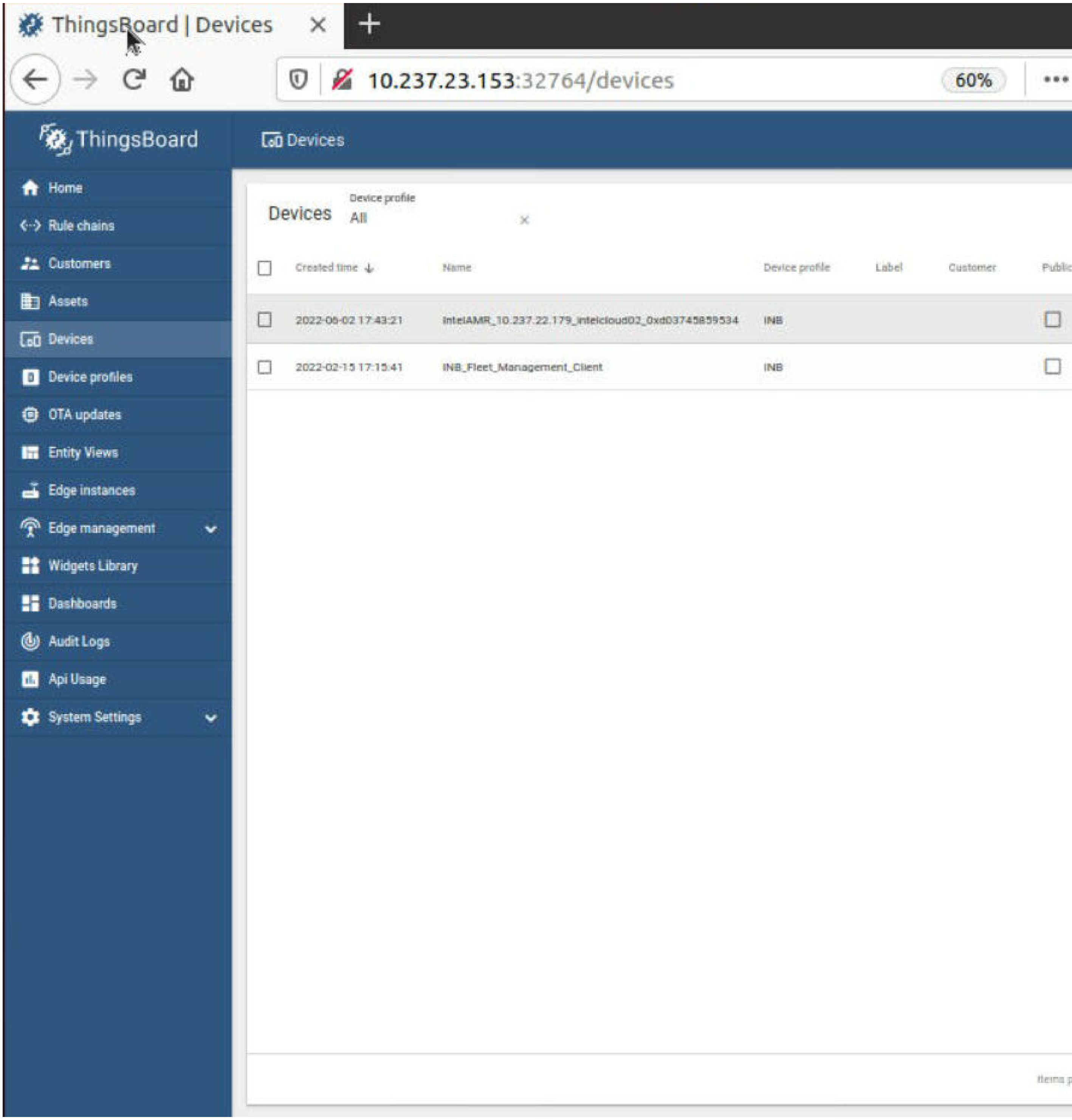
11. **Machine B- terminal 1:** In the client messages, verify that FDO completed.

```
amr-fdo-client         | ------------------------------------ TO2 Successful
--------------------------------------
amr-fdo-client         |
amr-fdo-client         | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
amr-fdo-client         | @FIDO Device Onboard Complete@
amr-fdo-client         | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
amr-fdo-client exited with code 0
```

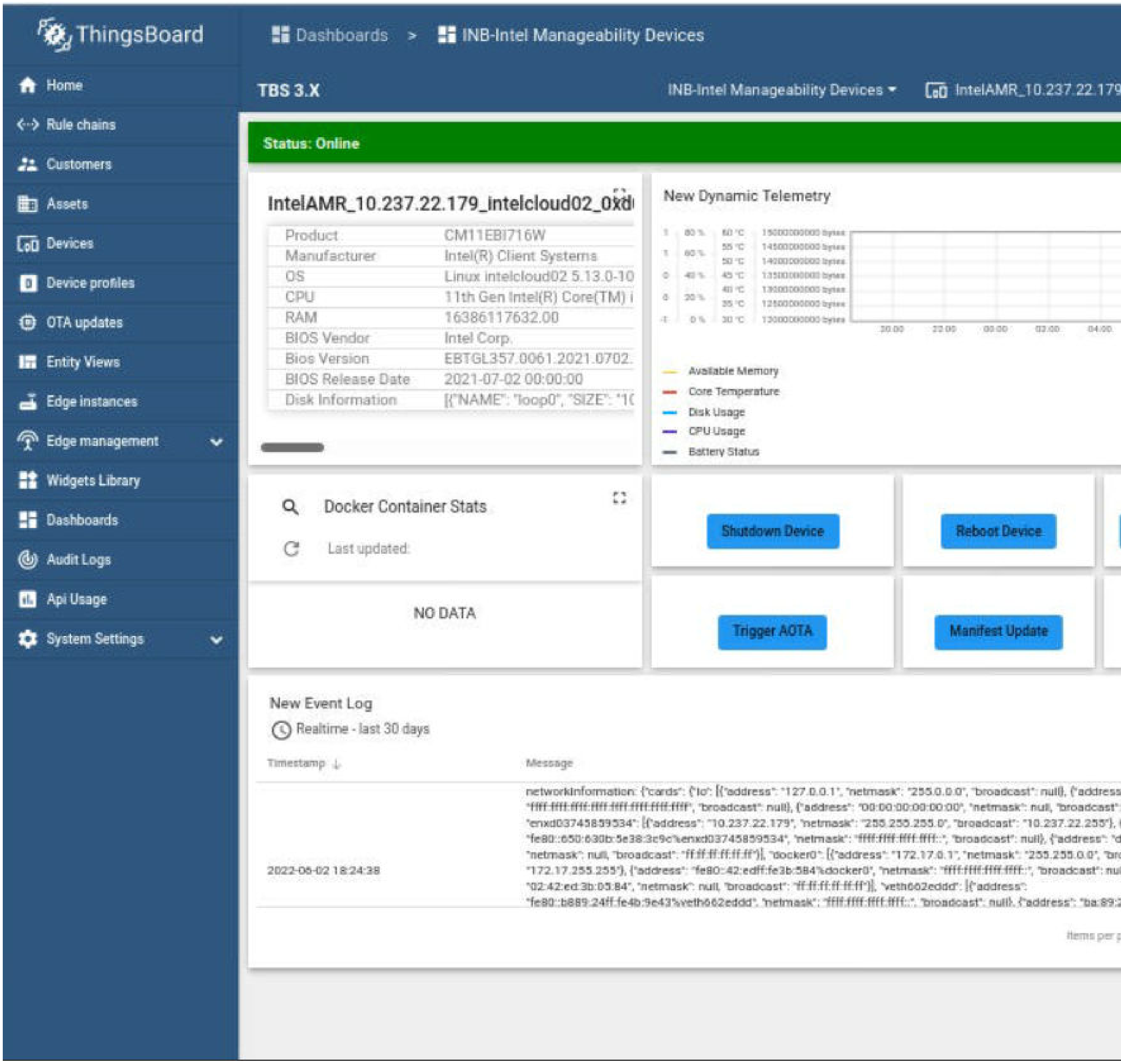
NOTE FDO protocol steps TO1 and TO2 can take more than five minutes.

Expected result:

Machine A-2: In the ThingsBoard* GUI, Machine B was added in Devices as a new device.



NOTE The device is online on the Dashboard after the Intel® In-Band Manageability container in Machine B is automatically brought up successfully.



Machine B: The wandering app is deployed from the Intel® Smart Edge Open controller, and the robot starts to wander around.

Hosts Cleanup

1. **Machine B- terminal 1:**

```
docker-compose -f 01_docker_sdk_env/docker_compose/01_amr/amr-sdk.all.yml down
```

2. **Machine B- terminal 2:**

```
docker-compose -f 01_docker_sdk_env/docker_compose/02_edge_server/edge-server.all.yml down
```

3. **Machine C- terminal 1:**

```
docker-compose -f 01_docker_sdk_env/docker_compose/02_edge_server/edge-server.all.yml down
```

4. Remove the device in the ThingsBoard* web interface.
5. **Machine B- terminal 3:** If the fleet-management container is already running:

```
docker rm -f $(docker ps -aq --filter name=amr-fleet-management)
```

> **NOTE** If you get an error message, verify that the fleet management container is not running and that the amr-fleet-management tag is `latest`. For example:
>
> ```
> amr-fleet-management:latest
> ```

Known Issues and Limitations

Only `amr-fleet-management:latest` is supported.

Troubleshooting

- Verify that the MQTT service is running:

```
systemctl status mosquitto.service
```

  If the command above returns `Active: failed`:

```
chmod -R 755 /etc/mosquitto/
systemctl restart mosquitto.service
systemctl status mosquitto.service
```

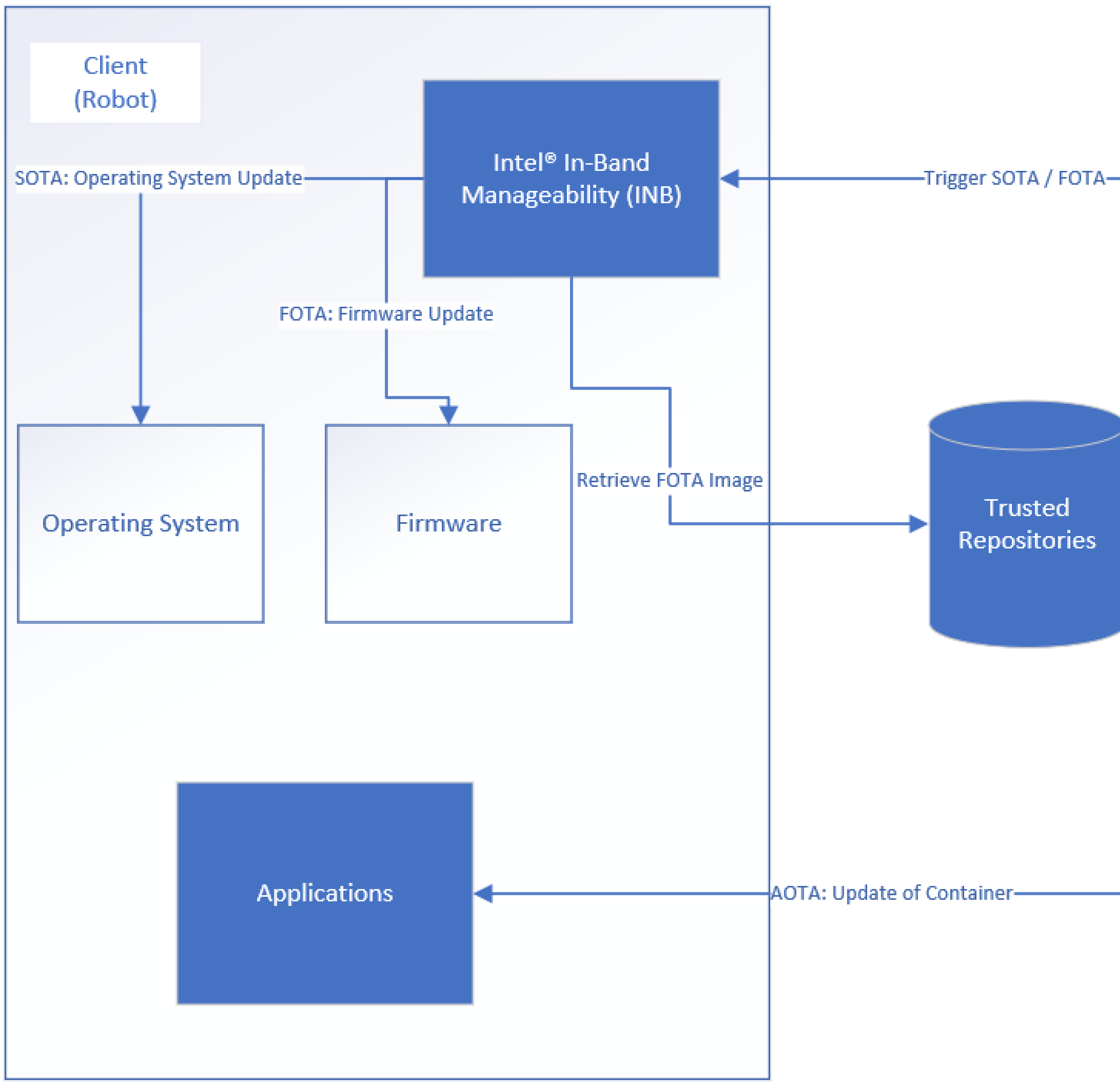
  Expected result: The status of the mosquitto service is `Active: active`
- For detailed EI for AMR target installation steps, see the Get Started Guide for Robots.

FDO References

Term	Reference
DMS	N/A
FDO	https://fidoalliance.org/intro-to-fido-device-onboard/
FIDO	https://en.wikipedia.org/wiki/FIDO_Alliance
RV	https://fidoalliance.org/specs/FDO/FIDO-Device-Onboard-RD-v1.0-20201202.html
Intel® SDO	https://www.intel.com/content/www/us/en/internet-of-things/secure-device-onboard.html

OTA Updates

The OTA updates solution is based on the Basic Fleet Management architecture. The following diagram presents the architecture, components, and communications for these use cases.



Robot Prerequisites

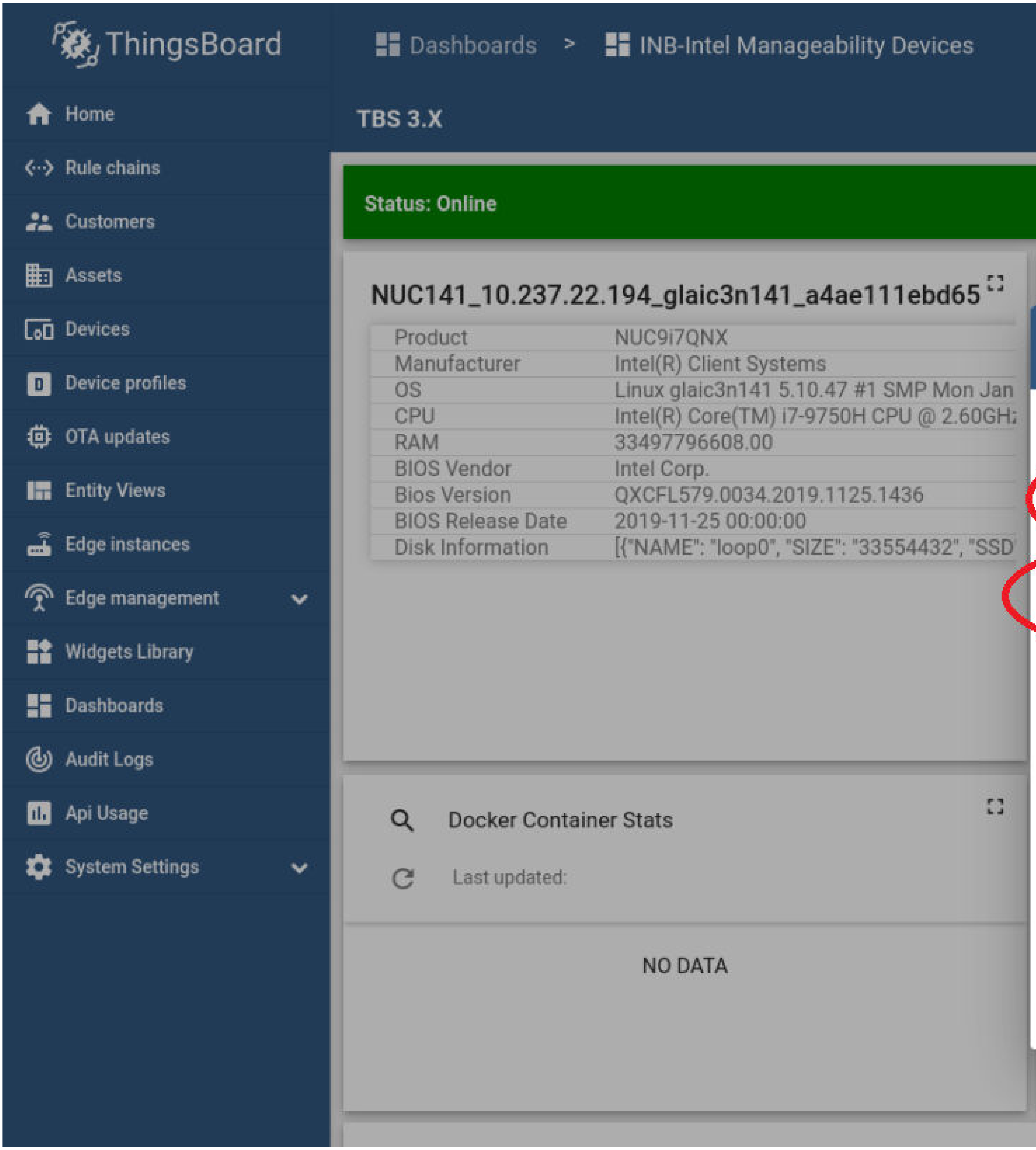
Run the following command on the client host:

```
sudo apt-get update && DEBIAN_FRONTEND=noninteractive  sudo apt-get install --no-install-
recommends -q -y  \
software-properties-
common                                                                                  \
```

```
&& if [[ -z "$http_proxy" ]] ; then  sudo apt-key adv --keyserver
keys.gnupg.net                           \
--recv-key
F6E65AC044F831AC80A06380C8B3A55A6F3EFCDE;                                                        \
else sudo apt-key adv --keyserver keys.gnupg.net --keyserver-
options                                          \
http-proxy="${http_proxy}" --recv-key F6E65AC044F831AC80A06380C8B3A55A6F3EFCDE;
fi                         \
|| if [[ -z "$http_proxy" ]] ; then sudo apt-key adv --keyserver hkp://
keyserver.ubuntu.com:80             \
--recv-key F6E65AC044F831AC80A06380C8B3A55A6F3EFCDE;
else                                                  \
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --keyserver-
options                            \
http-proxy="${http_proxy}" --recv-key F6E65AC044F831AC80A06380C8B3A55A6F3EFCDE;
fi                         \
&& sudo add-apt-repository "deb https://librealsense.intel.com/Debian/apt-repo focal main" -
u              \
&& DEBIAN_FRONTEND=noninteractive sudo apt-get install --no-install-recommends -q -
y                       \
rsync
      \
librealsense2=2.50.*                                                    \
librealsense2-utils=2.50.*                                              \
librealsense2-dev=2.50.*                                                \
librealsense2-gl=2.50.*                                                 \
librealsense2-net=2.50.*                                                \
librealsense2-dbg=2.50.*                                                \
librealsense2-udev-rules=2.50.*                                         \
&& sudo rm -rf /var/lib/apt/lists/*
```

Operating System Update

On the ThingsBoard* dashboard, click **Trigger SOTA**, select **update**, and click **SEND**. Update progress is visible in the ThingsBoard logs. The client host reboots after SOTA completes.


ThingsBoard
Dashboards > INB-Intel Manageability Devices
Home
Rule chains
Customers
Assets
Devices
Device profiles
OTA updates
Entity Views
Edge instances
Edge management
Widgets Library
Dashboards
Audit Logs
Api Usage
System Settings
TBS 3.X
Status: Online
NUC141_10.237.22.194_glaic3n141_a4ae111ebd65
Product NUC9i7QNX
Manufacturer Intel(R) Client Systems
OS Linux glaic3n141 5.10.47 #1 SMP Mon Jan
CPU Intel(R) Core(TM) i7-9750H CPU @ 2.60GH
RAM 33497796608.00
BIOS Vendor Intel Corp.
Bios Version QXCFL579.0034.2019.1125.1436
BIOS Release Date 2019-11-25 00:00:00
Disk Information [{"NAME": "loop0", "SIZE": "33554432", "SSD
Docker Container Stats
Last updated:
NO DATA

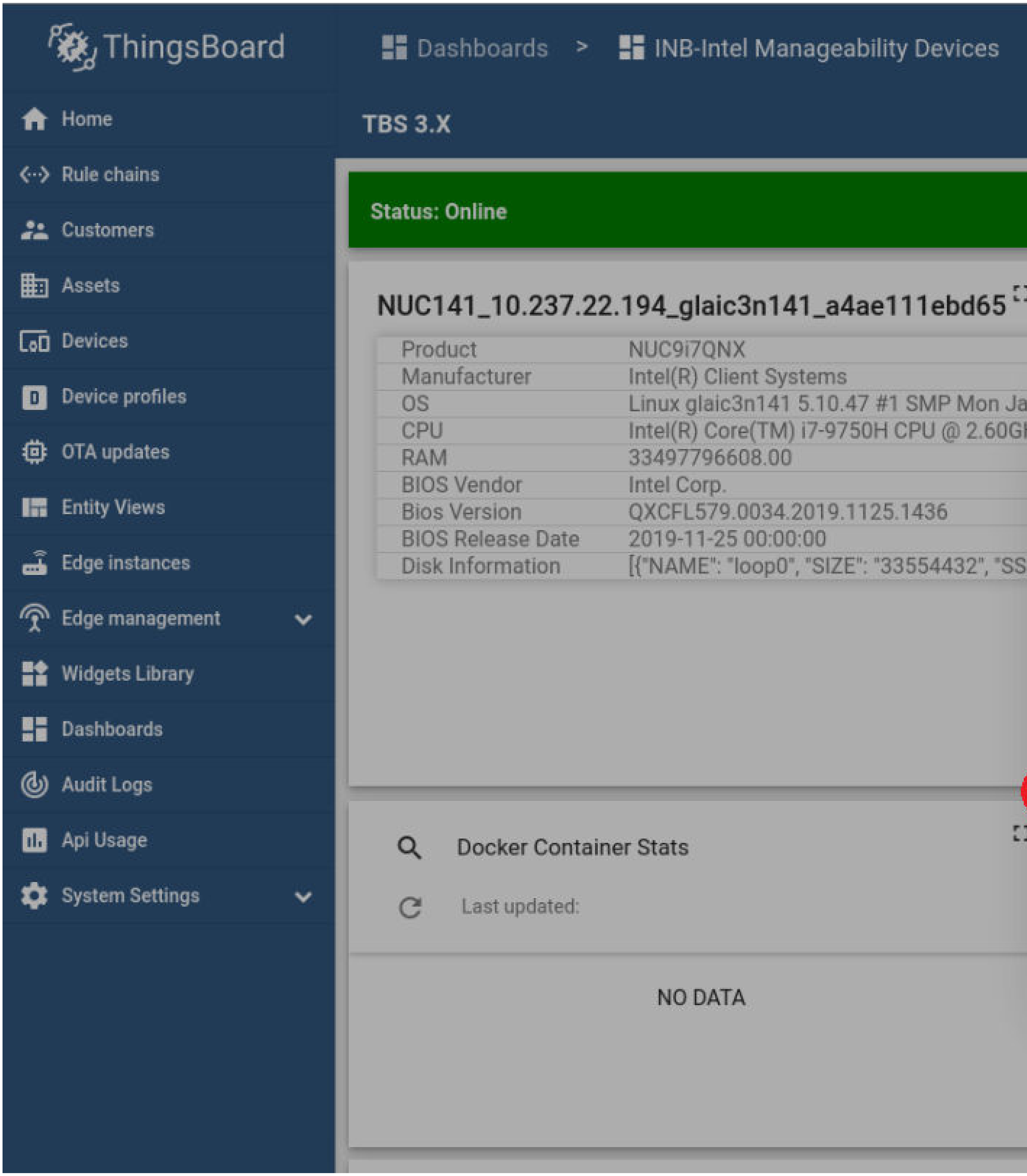
NOTE If the operating system update fails, `dpkg` may have been interrupted in the past or the SOTA cache directory is missing on the robot. Run the following commands to solve the issue:

```
sudo dpkg --configure -a
sudo mdkir -p /var/cache/manageability/repository-tool/sota
```

Firmware Update

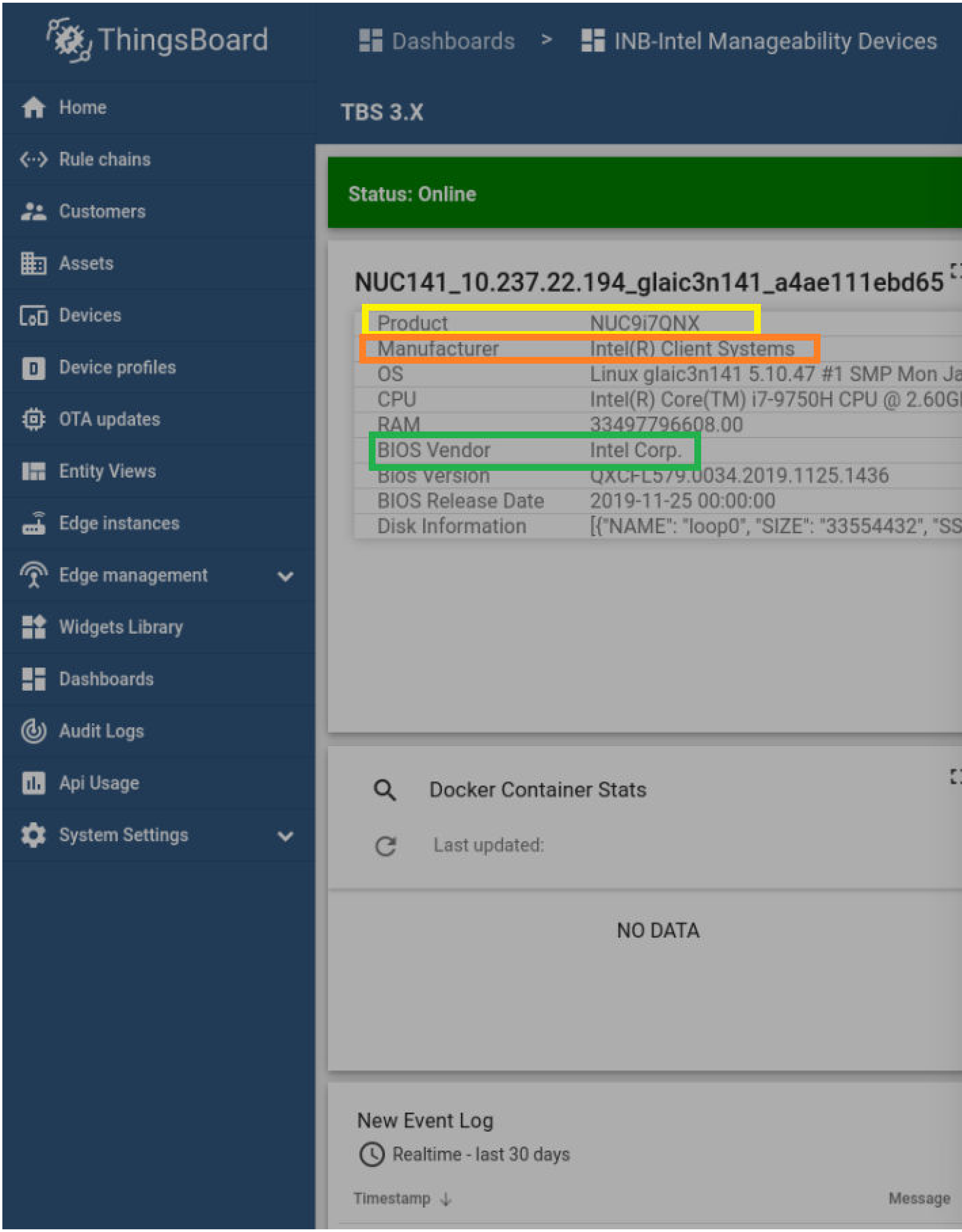
This example updates the Intel® RealSense™ camera firmware.

1. Preparation for the Intel® RealSense™ camera firmware update:
    a. Download the firmware from https://dev.intelrealsense.com/docs/firmware-releases.
    b. Place the `.bin` file that contains the firmware in a `.tar.gz` archive. Make sure that you do not archive the entire directory, only the `.bin` file.
    c. Set up a basic HTTP server, and upload the `.tar.gz` on it as a trusted repository server.
2. On the ThingsBoard* dashboard, click **Trigger Config Update**.
3. Choose:
    - Command: append
    - Path: trustedRepositories:http://url-to-http-server/and-optional-path-if-necessary/



4. Click **Send**, and observe the logs in the ThingsBoard* bottom screen.
5. Trigger the firmware update.
   - BIOS Version: any number

- Fetch: http://url-to-http-server/and-optional-path-if-necessary/arhive-with-firmware.tar.gz
- Manufacturer: set the value according to the following image
- Product: set the value according to the following image
- Release Date: the current date in the YYYY-MM-DD format
- Vendor: set the value according to the following image
- Server Username and Server Password: only used if the HTTP server is password protected


ThingsBoard
Dashboards > INB-Intel Manageability Devices
Home
Rule chains
Customers
Assets
Devices
Device profiles
OTA updates
Entity Views
Edge instances
Edge management
Widgets Library
Dashboards
Audit Logs
Api Usage
System Settings
TBS 3.X
Status: Online
NUC141_10.237.22.194_glaic3n141_a4ae111ebd65
Product NUC9i7QNX
Manufacturer Intel(R) Client Systems
OS Linux glaic3n141 5.10.47 #1 SMP Mon Ja
CPU Intel(R) Core(TM) i7-9750H CPU @ 2.60G
RAM 33497796608.00
BIOS Vendor Intel Corp.
Bios Version QXCFL579.0034.2019.1125.1436
BIOS Release Date 2019-11-25 00:00:00
Disk Information [{"NAME": "loop0", "SIZE": "33554432", "SS
Docker Container Stats
Last updated:
NO DATA
New Event Log
Realtime - last 30 days
Timestamp
Message

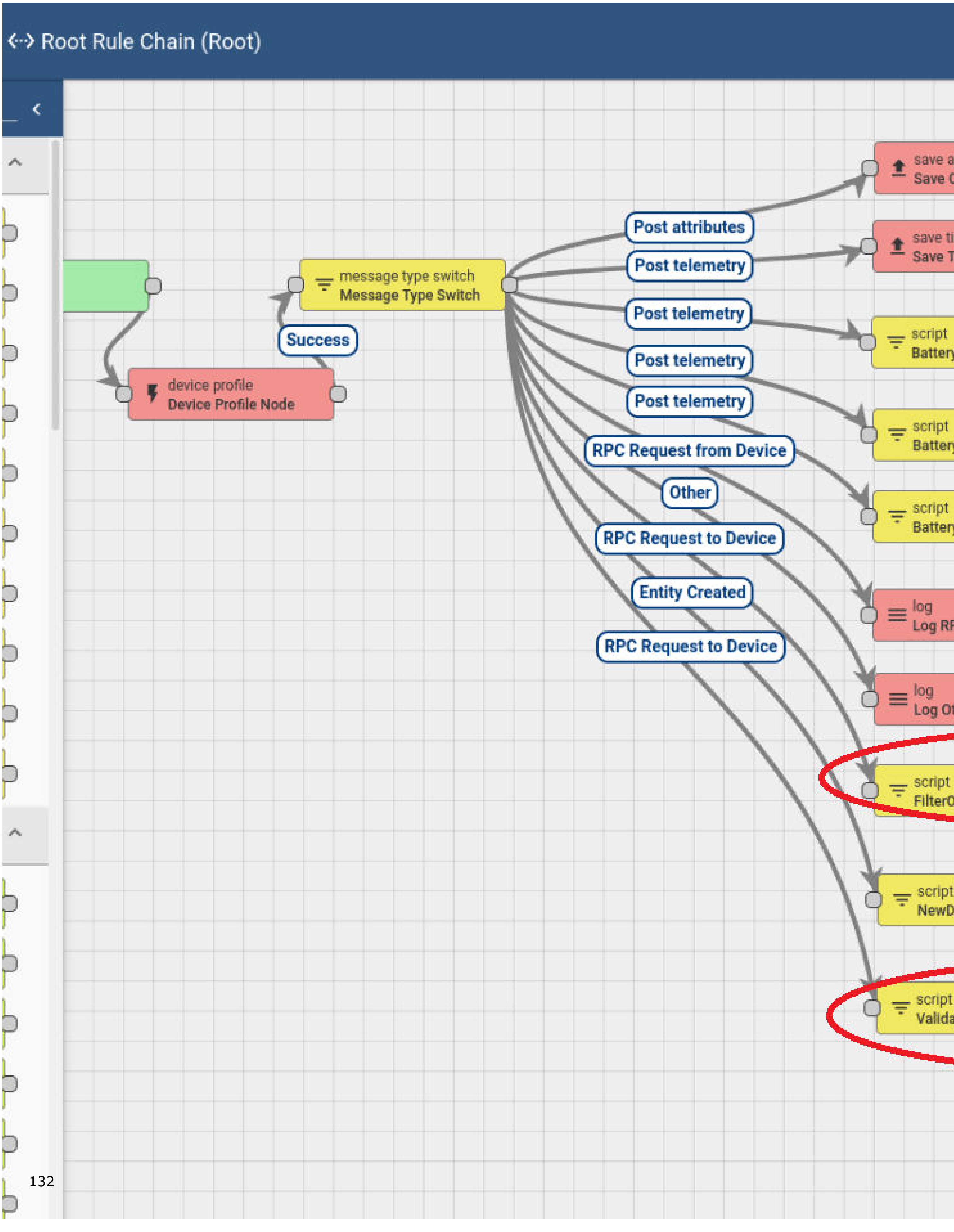
6. Click **Send**, and observe the logs in the ThingsBoard* bottom screen.

The client host reboots after the update complete.

Container Update

Because the containers (or applications) are orchestrated and deployed by Intel® Smart Edge Open, the ThingsBoard* Rule Chain routes AOTA trigger requests to the Intel® Smart Edge Open server.

The wandering application is used to present this use case.


Root Rule Chain (Root)
message type switch
Message Type Switch
Success
device profile
Device Profile Node
Post attributes
Post telemetry
Post telemetry
Post telemetry
Post telemetry
RPC Request from Device
Other
RPC Request to Device
Entity Created
RPC Request to Device
script
script
script
log
log
script
script
script

1. To create a new version of a Helm* chart, include the necessary modifications, and increment the version field from `Chart.yaml` in the Helm* charts:

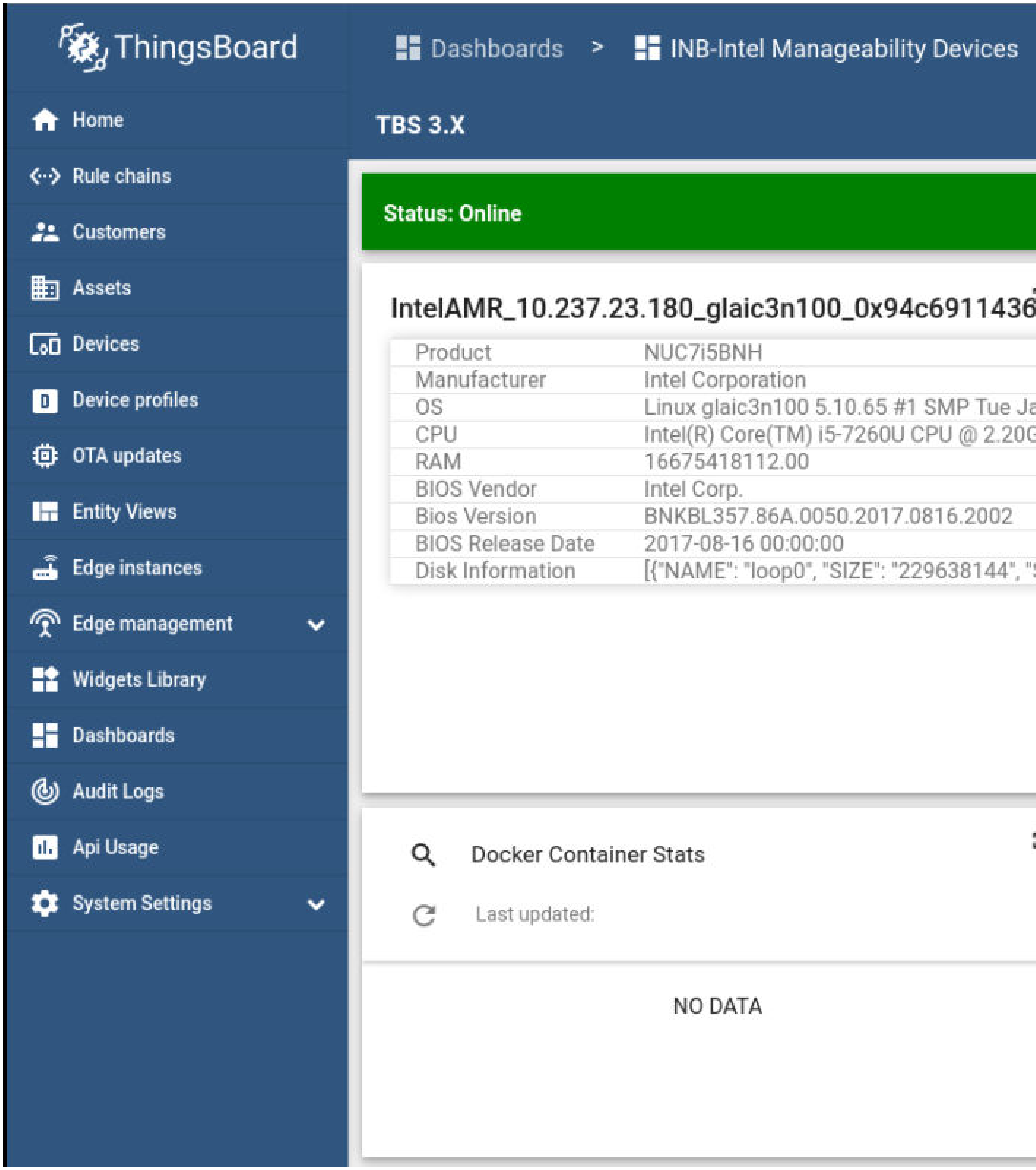
   Example:

```
$ cat helm_wandering/Chart.yaml
apiVersion: v2
appVersion: 1.0.0
description: A helm chart for wandering
name: wandering
type: application
version: 0.1.0
root@glaic3edge02:

$ cat helm_wandering_new/Chart.yaml
apiVersion: v2
appVersion: 1.0.0
description: A helm chart for wandering
name: wandering
type: application
version: 0.1.10
```

   NOTE Run the following copy command before triggering AOTA:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_server_containers/
cp 01_docker_sdk_env/docker_orchestration/ansible-playbooks/02_edge_server/wandering/
helm_wandering/values.yaml 01_docker_sdk_env/docker_orchestration/ansible-playbooks/
02_edge_server/wandering/helm_wandering_new/
```

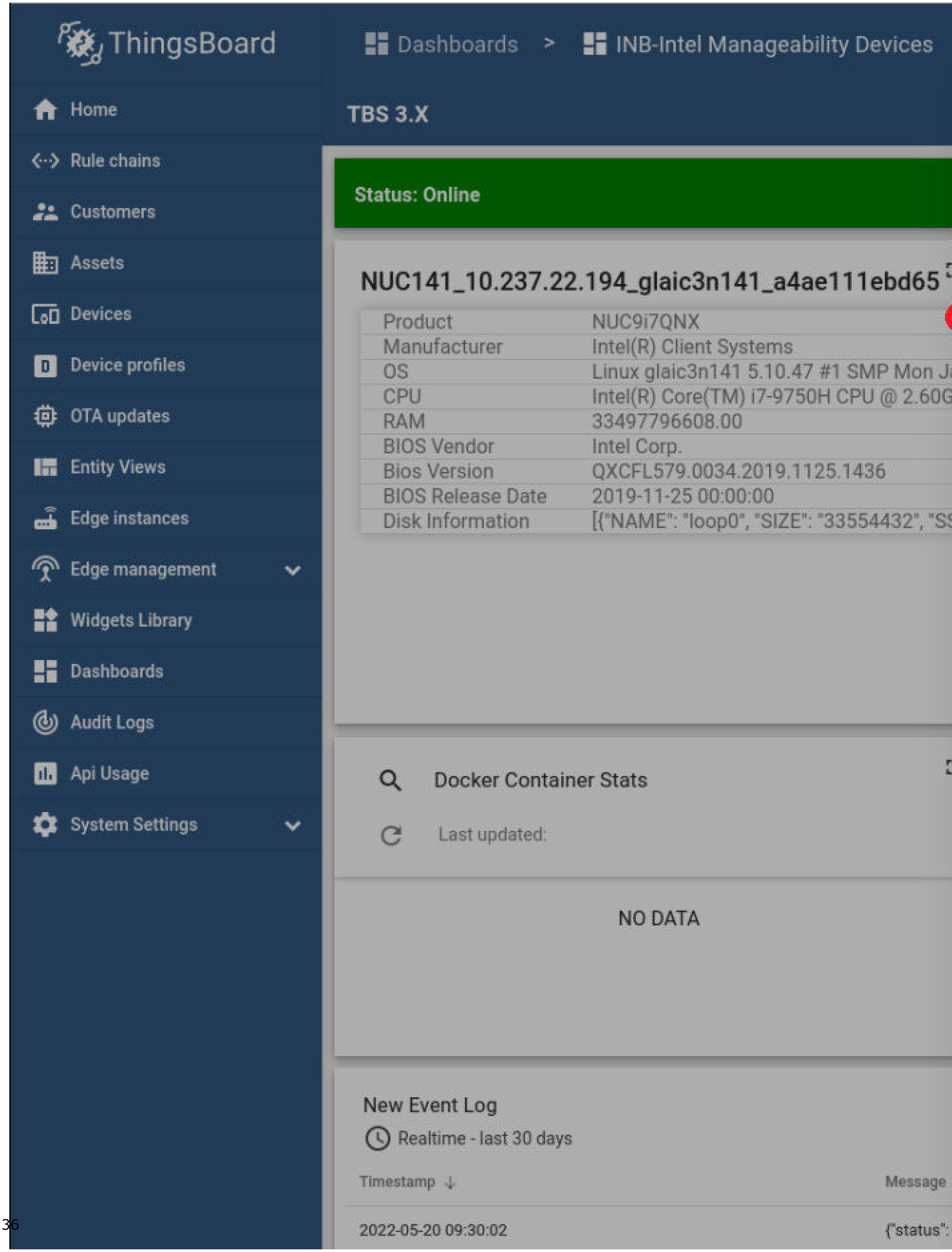
2. On the ThingsBoard* dashboard, click **Trigger AOTA**.



3. Select:

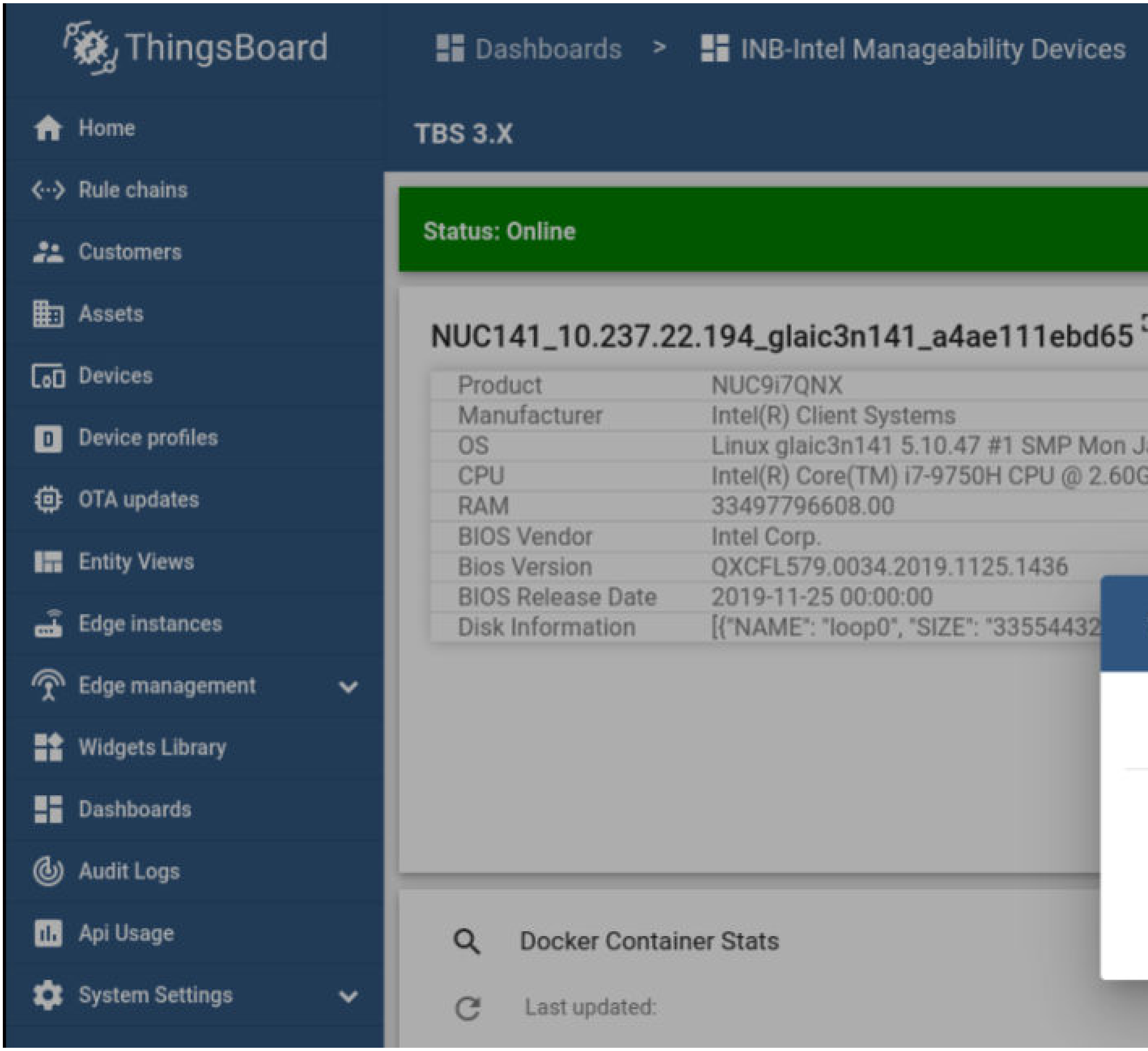
- App: application as App
- Command: update
- Container Tag: wandering

- Version:


ThingsBoard
Dashboards > INB-Intel Manageability Devices
Home
Rule chains
Customers
Assets
Devices
Device profiles
OTA updates
Entity Views
Edge instances
Edge management
Widgets Library
Dashboards
Audit Logs
Api Usage
System Settings
TBS 3.X
Status: Online
NUC141_10.237.22.194_glaic3n141_a4ae111ebd65
Product NUC9i7QNX
Manufacturer Intel(R) Client Systems
OS Linux glaic3n141 5.10.47 #1 SMP Mon J
CPU Intel(R) Core(TM) i7-9750H CPU @ 2.60G
RAM 33497796608.00
BIOS Vendor Intel Corp.
Bios Version QXCFL579.0034.2019.1125.1436
BIOS Release Date 2019-11-25 00:00:00
Disk Information [{"NAME": "loop0", "SIZE": "33554432", "S
Docker Container Stats
Last updated:
NO DATA
New Event Log
Realtime - last 30 days
Timestamp
Message
2022-05-20 09:30:02
{"status":

4. Click **Send**, and observe the logs in the ThingsBoard* bottom screen.

Known issue: A timeout error occurs after clicking **Send** because the AOTA request message is rerouted to the Intel® Smart Edge Open server, and the robot never receives the request.



Expected results:

- On the Intel® Smart Edge Open server, the Helm* chart version is renewed to a newer version.

```
helm list --all-namespaces
```

```
root@glaic3srv09:~/applications.robotics.mobile.container# helm list --all-namespaces
NAME            NAMESPACE               REVISION        UPDATED
cadvisor        telemetry               1               2022-05-12 13:40:02.956037814 +000
cert-manager    cert-manager            1               2022-05-12 13:33:01.782935656 +000
collectd        telemetry               1               2022-05-12 13:39:57.564988516 +000
fleet           fleet-management        7               2022-05-16 13:36:23.531631059 +000
harbor-app      harbor                  1               2022-05-12 13:34:24.300742013 +000
nfd-release     smartedge-system        1               2022-05-12 13:38:25.660145295 +000
prometheus      telemetry               1               2022-05-12 13:39:33.191894511 +000
statsd-exporter telemetry               1               2022-05-12 13:39:50.478831347 +000
wandering       wandering               8               2022-05-19 11:09:14.209028411 +000
root@glaic3srv09:~/applications.robotics.mobile.container# helm list --all-namespaces
NAME            NAMESPACE               REVISION        UPDATED
cadvisor        telemetry               1               2022-05-12 13:40:02.956037814 +000
cert-manager    cert-manager            1               2022-05-12 13:33:01.782935656 +000
collectd        telemetry               1               2022-05-12 13:39:57.564988516 +000
fleet           fleet-management        7               2022-05-16 13:36:23.531631059 +000
harbor-app      harbor                  1               2022-05-12 13:34:24.300742013 +000
nfd-release     smartedge-system        1               2022-05-12 13:38:25.660145295 +000
prometheus      telemetry               1               2022-05-12 13:39:33.191894511 +000
statsd-exporter telemetry               1               2022-05-12 13:39:50.478831347 +000
wandering       wandering               9               2022-05-20 09:20:25.305567035 +000
```

- On the EI for AMR device, the wandering app is updated.

Wandering Application Deployment

This tutorial describes how to deploy the wandering pods on a control plane and how to verify that the wandering containers are deployed successfully on the EI for AMR labeled nodes.

- Machine A is the Intel® Smart Edge Open control plane.
- Machine B is the device added using Device Onboarding End-to-End Use Case

Prerequisites: Intel® Smart Edge Open multi-node deployment configured per the Get Started Guide for Robots.

1. **On Machine A**:

```
ansible-playbook --extra-vars '{ "pod_replicas":6}' AMR_server_containers/01_docker_sdk_env/
docker_orchestration/ansible-playbooks/02_edge_server/wandering/wandering_install.yaml
```

> **NOTE** The number of `pod_replicas` is up to you.

Expected result: After installing the wandering playbook, a pod is deployed on the existing node that is labeled EI for AMR - one pod per node. The remaining pods remain in a pending state and are deployed when a new node labeled EI for AMR is added to the Intel® Smart Edge Open cluster.

2. **On Machine A**, verify that services, pods, and deployment are running:

```
$ kubectl get all  --output=wide --namespace wandering
 NAME                                          READY   STATUS        RESTARTS      AGE
IP                NODE              NOMINATED NODE   READINESS GATES
 pod/wandering-deployment-57764cb49c-d2txw    0/8     Pending       0             17h
<none>            <none>            <none>           <none>
 pod/wandering-deployment-57764cb49c-gsh5x    0/8     Pending       0             17h
<none>            <none>            <none>           <none>
 pod/wandering-deployment-57764cb49c-kk6qw    0/8     Pending       0             17h
<none>            <none>            <none>           <none>
 pod/wandering-deployment-57764cb49c-sc4lt    0/8     Pending       0             17h
<none>            <none>            <none>           <none>
```

```
 pod/wandering-deployment-74cf696fdb-cwh5n   0/8     Pending       0             17h
<none>           <none>           <none>           <none>
 pod/wandering-deployment-74cf696fdb-spr5v   0/8     Pending       0             17h
<none>           <none>           <none>           <none>

 NAME                          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE   SELECTOR
 service/wandering-service   ClusterIP   10.103.4.38      <none>        80/TCP    22h
app.kubernetes.io/instance=wandering-abcxzy,app.kubernetes.io/name=wandering

 NAME                                   READY   UP-TO-DATE   AVAILABLE   AGE
CONTAINERS

IMAGES



                                                SELECTOR
 deployment.apps/wandering-deployment   1/6     4            1           22h   amr-realsense,amr-
ros-base-camera-tf,amr-aaeon-amr-interface,amr-ros-base-teleop,amr-collab-slam,amr-
fastmapping,amr-nav2,amr-wandering   10.237.23.153:30003/intel/amr-
realsense:latest,10.237.23.153:30003/intel/amr-aaeon-amr-interface:latest,10.237.23.153:30003/
intel/amr-ros-base:latest,10.237.23.153:30003/intel/amr-collab-slam:latest,10.237.23.153:30003/
intel/amr-fastmapping:latest,10.237.23.153:30003/intel/amr-nav2:latest,10.237.23.153:30003/intel/
amr-wandering:latest   app.kubernetes.io/instance=wandering-abcxzy,app.kubernetes.io/
name=wandering

 NAME                                              DESIRED   CURRENT   READY   AGE
CONTAINERS

IMAGES



                                                SELECTOR
 replicaset.apps/wandering-deployment-57764cb49c   4         4         0       20h   amr-
realsense,amr-ros-base-camera-tf,amr-aaeon-amr-interface,amr-ros-base-teleop,amr-collab-slam,amr-
fastmapping,amr-nav2,amr-wandering   10.237.23.153:30003/intel/amr-
realsense:2022.2,10.237.23.153:30003/intel/amr-ros-base-camera-tf:2022.2,10.237.23.153:30003/
intel/amr-aaeon-amr-interface:2022.2,10.237.23.153:30003/intel/amr-ros-base-
teleop:2022.2,10.237.23.153:30003/intel/amr-collab-slam:2022.2,10.237.23.153:30003/intel/amr-
fastmapping:2022.2,10.237.23.153:30003/intel/amr-nav2:2022.2,10.237.23.153:30003/intel/amr-
wandering:2022.2   app.kubernetes.io/instance=wandering-abcxzy,app.kubernetes.io/
name=wandering,pod-template-hash=57764cb49c
 replicaset.apps/wandering-deployment-74cf696fdb   3         3         1       22h   amr-
realsense,amr-ros-base-camera-tf,amr-aaeon-amr-interface,amr-ros-base-teleop,amr-collab-slam,amr-
fastmapping,amr-nav2,amr-wandering   10.237.23.153:30003/intel/amr-
realsense:2022.2,10.237.23.153:30003/intel/amr-ros-base-camera-tf:2022.2,10.237.23.153:30003/
intel/amr-aaeon-amr-interface:2022.2,10.237.23.153:30003/intel/amr-ros-base-
teleop:2022.2,10.237.23.153:30003/intel/amr-collab-slam:2022.2,10.237.23.153:30003/intel/amr-
fastmapping:2022.2,10.237.23.153:30003/intel/amr-nav2:2022.2,10.237.23.153:30003/intel/amr-
wandering:2022.2   app.kubernetes.io/instance=wandering-abcxzy,app.kubernetes.io/
name=wandering,pod-template-hash=74cf696fdb
```

3. If EI for AMR labeled nodes are in the Intel® Smart Edge Open cluster or after Device Onboarding End-to-End Use Case is completed, the followings logs are displayed on the Intel® Smart Edge Open control plane on Machine A:

```
NAME                                            READY   STATUS    RESTARTS        AGE
IP             NODE             NOMINATED NODE   READINESS GATES
pod/wandering-deployment-86d6b669d6-27rcd   0/8     Pending   0               7m34s
<none>         <none>           <none>           <none>
pod/wandering-deployment-86d6b669d6-cbd2x   0/8     Pending   0               7m34s
```

```
<none>         <none>           <none>           <none>
pod/wandering-deployment-86d6b669d6-rwbnd   0/8     Pending   0               7m34s
<none>         <none>           <none>           <none>
pod/wandering-deployment-86d6b669d6-rzlgr   8/8     Running   1 (7m33s ago)   7m34s
10.245.188.4   intelcloudgl05   <none>           <none>
pod/wandering-deployment-86d6b669d6-tnp4z   0/8     Pending   0               7m34s
<none>         <none>           <none>           <none>
pod/wandering-deployment-86d6b669d6-vjpx9   0/8     Pending   0               7m34s
<none>         <none>           <none>           <none>

NAME                        TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE     SELECTOR
service/wandering-service   ClusterIP   10.103.4.38   <none>        80/TCP    7m35s
app.kubernetes.io/instance=wandering-abcxzy,app.kubernetes.io/name=wandering

NAME                                   READY   UP-TO-DATE   AVAILABLE   AGE
CONTAINERS

IMAGES




      SELECTOR
deployment.apps/wandering-deployment   1/6     6            1           7m35s   amr-
realsense,amr-ros-base-camera-tf,amr-aaeon-amr-interface,amr-ros-base-teleop,amr-collab-slam,amr-
fastmapping,amr-nav2,amr-wandering   10.237.22.39:30003/intel/amr-
realsense:2022.2,10.237.22.39:30003/intel/amr-ros-base-camera-tf:2022.2,10.237.22.39:30003/intel/
amr-aaeon-amr-interface:2022.2,10.237.22.39:30003/intel/amr-ros-base-
teleop:2022.2,10.237.22.39:30003/intel/amr-collab-slam:2022.2,10.237.22.39:30003/intel/amr-
fastmapping:2022.2,10.237.22.39:30003/intel/amr-nav2:2022.2,10.237.22.39:30003/intel/amr-
wandering:2022.2   app.kubernetes.io/instance=wandering-abcxzy,app.kubernetes.io/name=wandering

NAME                                              DESIRED   CURRENT   READY   AGE
CONTAINERS

IMAGES




      SELECTOR
replicaset.apps/wandering-deployment-86d6b669d6   6         6         1       7m35s   amr-
realsense,amr-ros-base-camera-tf,amr-aaeon-amr-interface,amr-ros-base-teleop,amr-collab-slam,amr-
fastmapping,amr-nav2,amr-wandering   10.237.22.39:30003/intel/amr-
realsense:2022.2,10.237.22.39:30003/intel/amr-ros-base-camera-tf:2022.2,10.237.22.39:30003/intel/
amr-aaeon-amr-interface:2022.2,10.237.22.39:30003/intel/amr-ros-base-
teleop:2022.2,10.237.22.39:30003/intel/amr-collab-slam:2022.2,10.237.22.39:30003/intel/amr-
fastmapping:2022.2,10.237.22.39:30003/intel/amr-nav2:2022.2,10.237.22.39:30003/intel/amr-
wandering:2022.2   app.kubernetes.io/instance=wandering-abcxzy,app.kubernetes.io/
name=wandering,pod-template-hash=86d6b669d6
```

4. Verify that the Docker* images are present on Machine B:

```
$ docker images

 <MACHINE_A_IP>:30003/intel/amr-ros-base-camera-tf                      latest
31735754089b   2 days ago      8.25GB
 <MACHINE_A_IP>:30003/intel/amr-wandering                               latest
31735754089b   2 days ago      8.25GB
 <MACHINE_A_IP>:30003/intel/amr-fastmapping                             latest
5c1bbefc1d17   2 days ago      2.28GB
 <MACHINE_A_IP>:30003/intel/amr-collab-slam                             latest
```

```
415975276b1f   2 days ago      3.24GB
 <MACHINE_A_IP>:30003/intel/amr-aaeon-amr-interface                      latest
5d94f57da0d1   2 days ago      2.37GB
 <MACHINE_A_IP>:30003/intel/amr-realsense                                latest
1dab67f4d287   2 days ago      3GB
 <MACHINE_A_IP>:30003/intel/amr-ros-base-camera-tf                       latest
0ac635f5633f   2 days ago      1.76GB
 <MACHINE_A_IP>:30003/intel/amr-nav2                                     latest
769353e041bf   2 days ago      3.55GB
```

NOTE Pod deployment may take a while because of the size of the Docker* containers from the pod. If you get an error after the deployment, wait a few minutes. The pods automatically restart, and the error goes away. If the error persists after a few automatic restarts, restart the pod manually from **Machine A**:

```
$ kubectl rollout restart deployment wandering-deployment -n wandering
```

5. Verify that the Docker* container is running on Machine B:

```
$ docker ps

 CONTAINER ID   IMAGE                                              COMMAND
CREATED        STATUS        PORTS     NAMES
 86184dab6d92   10.237.22.39:30003/intel/amr-ros-base-camera-tf    "/bin/bash -c 'sourc…"
About a minute ago   Up About a minute             k8s_amr-ros-base-teleop_wandering-
deployment-86d6b669d6-rzlgr_wandering_c00ecd97-2217-4f4f-a62c-9f99bc44ac7d_1
 9d19c163076f   10.237.22.39:30003/intel/amr-wandering             "/bin/bash -c 'sourc…"
About a minute ago   Up About a minute             k8s_amr-wandering_wandering-
deployment-86d6b669d6-rzlgr_wandering_c00ecd97-2217-4f4f-a62c-9f99bc44ac7d_0
 b9f03850310e   10.237.22.39:30003/intel/amr-nav2                  "/bin/bash -c 'sourc…"
About a minute ago   Up About a minute             k8s_amr-nav2_wandering-deployment-86d6b669d6-
rzlgr_wandering_c00ecd97-2217-4f4f-a62c-9f99bc44ac7d_0
 8fb3fb882505   10.237.22.39:30003/intel/amr-fastmapping           "/bin/bash -c 'sourc…"
About a minute ago   Up About a minute             k8s_amr-fastmapping_wandering-
deployment-86d6b669d6-rzlgr_wandering_c00ecd97-2217-4f4f-a62c-9f99bc44ac7d_0
 1f122686f8e1   10.237.22.39:30003/intel/amr-collab-slam           "/bin/bash -c 'sourc…"
About a minute ago   Up About a minute             k8s_amr-collab-slam_wandering-
deployment-86d6b669d6-rzlgr_wandering_c00ecd97-2217-4f4f-a62c-9f99bc44ac7d_0
 ee7e6cd8b50a   10.237.22.39:30003/intel/amr-aaeon-amr-interface   "/bin/bash -c 'sourc…"
About a minute ago   Up About a minute             k8s_amr-aaeon-amr-interface_wandering-
deployment-86d6b669d6-rzlgr_wandering_c00ecd97-2217-4f4f-a62c-9f99bc44ac7d_0
 009efc5405af   10.237.22.39:30003/intel/amr-ros-base-camera-tf    "/bin/bash -c 'sourc…"
About a minute ago   Up About a minute             k8s_amr-ros-base-camera-tf_wandering-
deployment-86d6b669d6-rzlgr_wandering_c00ecd97-2217-4f4f-a62c-9f99bc44ac7d_0
 1a6409b8c361   10.237.22.39:30003/intel/amr-realsense             "/bin/bash -c 'sourc…"
About a minute ago   Up About a minute             k8s_amr-realsense_wandering-
deployment-86d6b669d6-rzlgr_wandering_c00ecd97-2217-4f4f-a62c-9f99bc44ac7d_0
```

Troubleshooting for Robot Orchestration Tutorials

Setting a Static IP

Depending on your network setup, there are multiple ways to set a static IP.

- In a home network, check your local router on how to set static IP on your device using your MAC address.
- In a corporate network, please check with your local IT on how to set a static IP on a system.

Another option is to set it from your system's Operating System. A good tutorial on how to set a static IP using netplan can be found here.

Remember to run `netplan apply` after you are finished with the configuration.

- Make sure you that your system has the correct date:

```
date
```

If the date is incorrect, contact your local support team for help setting the correct date and time.

- To find the gateway:

```
ip route | grep default
```

- To find the name servers, find your interface name and replace it below:

```
nmcli device show <interface name> | grep IP4.DNS
```

virtualenv Error

If the following error is displayed:

```
Virtualenv location:
Warning: There was an unexpected error while activating your virtualenv.
Continuing anyway…
Traceback (most recent call last):
File "./deploy.py", line 24, in <module>
from scripts import log_all
ImportError: cannot import name 'log_all' from 'scripts' (/home/
test/.local/lib/python3.8/site-packages/scripts/__init__.py)
```

Remove the `~/.local/lib/python3.8/` directory and run the following commands:

```
pip install --user -U pip
pip freeze --user | cut -d'=' -f1 | xargs pip install --user -U
```

termcolor Error

If the following error is displayed:

```
Failed to install termcolor. b'/usr/local/lib/python3.8/dist-packages/pkg_resources/
__init__.py:122:
```

```
python3 -m pip uninstall setuptools
python3 -m pip install setuptools
```

Restart the target and run:

```
python3 -m pip install --upgrade setuptools
```

Failed OpenSSL Download

If the following error is displayed:

```
FAILED - RETRYING: OpenSSL download from https://www.openssl.org/source/openssl-1.1.1i.tar.gz (4
retries left).
FAILED - RETRYING: OpenSSL download from https://www.openssl.org/source/openssl-1.1.1i.tar.gz (4
retries left).
FAILED - RETRYING: OpenSSL download from https://www.openssl.org/source/openssl-1.1.1i.tar.gz (3
retries left).
```

```
FAILED - RETRYING: OpenSSL download from https://www.openssl.org/source/openssl-1.1.1i.tar.gz (3
retries left).
FAILED - RETRYING: OpenSSL download from https://www.openssl.org/source/openssl-1.1.1i.tar.gz (2
retries left).
FAILED - RETRYING: OpenSSL download from https://www.openssl.org/source/openssl-1.1.1i.tar.gz (2
retries left).
FAILED - RETRYING: OpenSSL download from https://www.openssl.org/source/openssl-1.1.1i.tar.gz (1
retries left).
FAILED - RETRYING: OpenSSL download from https://www.openssl.org/source/openssl-1.1.1i.tar.gz (1
retries left)
```

Run the following commands:

```
wget --directory-prefix=/tmp http://certificates.intel.com/repository/certificates/
IntelSHA2RootChain-Base64.zip

sudo unzip -o /tmp/IntelSHA2RootChain-Base64.zip -d /usr/local/share/ca-certificates/

rm /tmp/IntelSHA2RootChain-Base64.zip

update-ca-certificates
```

“Isecl control plane IP not set” Error

If the following error is displayed:

```
TASK [Check control plane IP]
************************************************************************************************
************************************************************************************************
*************
task path: /root/dek/roles/security/isecl/common/tasks/precheck.yml:7
Wednesday 16 February 2022  15:36:34 +0000 (0:00:00.047)       0:00:05.373 ****
fatal: [node01]: FAILED! => {
   "changed": false
}

MSG:

Isecl control plane IP not set!
fatal: [node02]: FAILED! => {
   "changed": false
}

MSG:

Isecl control plane IP not set!
fatal: [controller]: FAILED! => {
   "changed": false
}

MSG:

Isecl control plane IP not set!
```

Update the `~/dek/inventory/default/group_vars/all/10-default.yml` file with:

```
# Install isecl attestation components (TA, ihub, isecl k8s controller and scheduler extension)
platform_attestation_node: false
```

“PCCS IP address not set” Error

If the following error is displayed:

```
TASK [Check PCCS IP address]
***************************************************************************************************
***************************************************************************************************
**************
task path: /root/dek/roles/infrastructure/provision_sgx_enabled_platform/tasks/
param_precheck.yml:7
Wednesday 16 February 2022  15:39:59 +0000 (0:00:00.060)       0:00:05.688 ****
fatal: [node01]: FAILED! => {
   "changed": false
}

MSG:

PCCS IP address not set!
fatal: [node02]: FAILED! => {
   "changed": false
}

MSG:

PCCS IP address not set!
fatal: [controller]: FAILED! => {
   "changed": false
}

MSG:

PCCS IP address not set!
```

Update the `~/dek/inventory/default/group_vars/all/10-default.yml` file with:

```
### Software Guard Extensions
# SGX requires kernel 5.11+, SGX enabled in BIOS and access to PCC service
sgx_enabled: false
```

“no supported NIC is selected” Error

If the following error is displayed:

```
sriovnetwork.sriovnetwork.openshift.io/sriov-vfio-network-c1p1 unchanged
STDERR:
Error from server (no supported NIC is selected by the nicSelector in CR sriov-netdev-net-c0p0):
error when creating "sriov-netdev-net-c0p0-sriov_network_node_policy.yml": admission webhook
"operator-webhook.sriovnetwork.openshift.io" denied the request: no supported NIC is selected by
the nicSelector in CR sriov-netdev-net-c0p0
Error from server (no supported NIC is selected by the nicSelector in CR sriov-netdev-net-c1p0):
error when creating "sriov-netdev-net-c1p0-sriov_network_node_policy.yml": admission webhook
"operator-webhook.sriovnetwork.openshift.io" denied the request: no supported NIC is selected by
the nicSelector in CR sriov-netdev-net-c1p0
Error from server (no supported NIC is selected by the nicSelector in CR sriov-vfio-pci-net-
c0p1): error when creating "sriov-vfio-pci-net-c0p1-sriov_network_node_policy.yml": admission
webhook "operator-webhook.sriovnetwork.openshift.io" denied the request: no supported NIC is
selected by the nicSelector in CR sriov-vfio-pci-net-c0p1
Error from server (no supported NIC is selected by the nicSelector in CR sriov-vfio-pci-net-
```

```
c1p1): error when creating "sriov-vfio-pci-net-c1p1-sriov_network_node_policy.yml": admission
webhook "operator-webhook.sriovnetwork.openshift.io" denied the request: no supported NIC is
selected by the nicSelector in CR sriov-vfio-pci-net-c1p1
```

Update the `~/dek/inventory/default/group_vars/all/10-default.yml` file with:

```
sriov_network_operator_enable: false

## SR-IOV Network Operator configuration
sriov_network_operator_configure_enable: false
```

"Unexpected templating type error"

If the following error is displayed:

```
MSG:
AnsibleError: Unexpected templating type error occurred on (# SPDX-License-Identifier: Apache-2.0
# Copyright (c) 2020 Intel Corporation
apiVersion: v1
kind: ConfigMap
metadata:
   name: grafana-datasources
   namespace: telemetry
   labels:
      grafana_datasource: '1'
data:
   prometheus-tls.yaml: |-
      apiVersion: 1
      datasources:
      - name: Prometheus-TLS
         access: proxy
         editable: true
         orgId: 1
         type: prometheus
         url: https://prometheus:9099
         withCredentials: true
         isDefault: true
         jsonData:
            tlsAuth: true
            tlsAuthWithCACert: true
         secureJsonData:
            tlsCACert: |
               {{ telemetry_root_ca_cert.stdout | trim | indent(width=13, indentfirst=False) }}
            tlsClientCert: |
               {{ telemetry_grafana_cert.stdout | trim | indent(width=13, indentfirst=False) }}
            tlsClientKey: |
               {{ telemetry_grafana_key.stdout | trim | indent(width=13, indentfirst=False) }}
         version: 1
         editable: false
): do_indent() got an unexpected keyword argument 'indentfirst'
```

Update the `~/dek/roles/telemetry/grafana/templates/prometheus-tls-datasource.yml` file with:

```
-              {{ telemetry_root_ca_cert.stdout | trim | indent(width=13, indentfirst=False) }}
+              {{ telemetry_root_ca_cert.stdout | trim | indent(width=13, first=False) }}
-              {{ telemetry_grafana_cert.stdout | trim | indent(width=13, indentfirst=False) }}
+              {{ telemetry_grafana_cert.stdout | trim | indent(width=13, first=False) }}
-              {{ telemetry_grafana_key.stdout | trim | indent(width=13, indentfirst=False) }}
+              {{ telemetry_grafana_key.stdout | trim | indent(width=13, first=False) }}
```

“Wait till all Harbor resources ready” Message

If the following log is displayed:

```
TASK [kubernetes/cni : Wait till all Harbor resources ready]
***************************************************************************************************
********************************************************************************
task path: /home/user/dek/roles/kubernetes/cni/tasks/main.yml:20
Tuesday 16 November 2021 14:41:58 +0100 (0:00:00.070) 0:04:39.646 ******
FAILED - RETRYING: Wait till all Harbor resources ready (60 retries left).
FAILED - RETRYING: Wait till all Harbor resources ready (59 retries left).
FAILED - RETRYING: Wait till all Harbor resources ready (58 retries left).
FAILED - RETRYING: Wait till all Harbor resources ready (57 retries left).
FAILED - RETRYING: Wait till all Harbor resources ready (56 retries left).
FAILED - RETRYING: Wait till all Harbor resources ready (55 retries left).
FAILED - RETRYING: Wait till all Harbor resources ready (54 retries left).
FAILED - RETRYING: Wait till all Harbor resources ready (53 retries left).
FAILED - RETRYING: Wait till all Harbor resources ready (52 retries left).
FAILED - RETRYING: Wait till all Harbor resources ready (51 retries left).
FAILED - RETRYING: Wait till all Harbor resources ready (50 retries left).
```

Wait approximately 30 minutes. The Intel® Smart Edge Open deployment script waits for the Harbor resources to be ready.

Installation Stuck

If the installation remains stuck with the following log:

```
TASK [infrastructure/os_setup : enable UFW]
***************************************************************************************************
**************************************************************************************************
task path: /root/dek/roles/infrastructure/os_setup/tasks/ufw_enable_debian.yml:12
Wednesday 16 February 2022  15:53:04 +0000 (0:00:01.627)       0:08:03.425 ****
NOTIFIED HANDLER reboot server for controller
changed: [controller] => {
   "changed": true,
   "commands": [
      "/usr/sbin/ufw status verbose",
      "/usr/bin/grep -h '^### tuple' /lib/ufw/user.rules /lib/ufw/user6.rules /etc/ufw/
user.rules /etc/ufw/user6.rules /var/lib/ufw/user.rules /var/lib/ufw/user6.rules",
      "/usr/sbin/ufw -f enable",
      "/usr/sbin/ufw status verbose",
      "/usr/bin/grep -h '^### tuple' /lib/ufw/user.rules /lib/ufw/user6.rules /etc/ufw/
user.rules /etc/ufw/user6.rules /var/lib/ufw/user.rules /var/lib/ufw/user6.rules"
   ]
}

MSG:

Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), deny (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW IN    Anywhere
22/tcp (v6)                ALLOW IN    Anywhere (v6)
```

Type `Ctrl-c`, and restart the installation. (Run the `./deploy.sh` script again.)

Pod Remains in "Terminating" State after Uninstall

After uninstall, if the pod does not stop but remains in "Terminating" state, enter the following commands:

```
kubectl get pods -n fleet-management
kubectl delete -n <pod_name_from_above_command> --grace-period=0 --force
ansible-playbook AMR_server_containers/01_docker_sdk_env/docker_orchestration/ansible-playbooks/
02_edge_server/fleet_management/fleet_management_playbook_uninstall.yaml
```

docker-compose Failure

If you see an error message that `docker-compose` fails with some variables not defined, add the environment variables to `.bashrc` so that they are available to all terminals:

```
export DOCKER_BUILDKIT=1
export COMPOSE_DOCKER_CLI_BUILD=1
export DOCKER_HOSTNAME=$(hostname)
export DOCKER_USER_ID=$(id -u)
export DOCKER_GROUP_ID=$(id -g)
export DOCKER_USER=$(whoami)
# Check with command
env | grep DOCKER
```

Keytool Not Installed

The keytool utility is used to create the certificate store. Install any preferred Java* version. For development, Intel used:

```
sudo apt install default-jre
# Check your Java version:
java -version
```

Corrupt Database or Nonresponsive Server

Reset the ThingsBoard* server with the following steps.

1. Uninstall the playbook:

```
ansible-playbook AMR_server_containers/01_docker_sdk_env/docker_orchestration/ansible-playbooks/
02_edge_server/fleet_management/fleet_management_playbook_uninstall.yaml
```

2. After uninstalling the playbook, wait several seconds for all fleet related containers to stop. Verify that there are no fleet containers running:

```
docker ps | grep fleet
```

3. Reinstall the playbook:

```
ansible-playbook AMR_server_containers/01_docker_sdk_env/docker_orchestration/ansible-playbooks/
02_edge_server/fleet_management/fleet_management_playbook_install.yaml
```

ThingsBoard* Server Errors

These errors can be fixed directly on the hosting machine using Docker* Compose. However, this requires automated steps using Ansible* playbooks, so try these fixes last.

- Reset the database to a pristine state (without customizations from Intel):

```
# delete database and start the server
# The state of server will be - without any customization from Intel.
sudo rm -rf ~/.mytb-data/db ~/.mytb-data/.firstlaunch ~/.mytb-data/.upgradeversion
docker-compose -f 01_docker_sdk_env/docker_compose/02_edge_server/edge-server.all.yml down
CHOOSE_USER=thingsboard docker-compose -f 01_docker_sdk_env/docker_compose/02_edge_server/edge-
server.all.yml up fleet-management
```

NOTE This only restarts the ThingsBoard* server, without Intel® Smart Edge Open.

- Reset the database to the preconfigured state (with customizations from Intel), and restart the server:

```
# Start the server with old/corrupted database
CHOOSE_USER=thingsboard docker-compose -f 01_docker_sdk_env/docker_compose/02_edge_server/edge-
server.all.yml up fleet-management
# attach to running container from another terminal:
docker exec -it  edge-server-sdk-fleet-management bash

# inside the container: replace the database with Intel-customized-database:
# Just press tb<tab>. The tb-server-reset-db.sh is present in /usr/local/bin folder, so it is
accessible from anywhere.
tb-server-reset-db.sh
# When asked press y and enter. Done.
# Now exit the container. and run below commands again to re-launch the server with
preconfigured-state of database (With Intel Customizations):
docker-compose -f 01_docker_sdk_env/docker_compose/02_edge_server/edge-server.all.yml down
CHOOSE_USER=thingsboard docker-compose -f 01_docker_sdk_env/docker_compose/02_edge_server/edge-
server.all.yml up fleet-management
```

NOTE This only restarts the ThingsBoard* server, without Intel® Smart Edge Open.

- When you deploy the ThingsBoard* container using Intel® Smart Edge Open Ansible* playbook, sometimes the server cannot start due to following error:

```
edge-server-sdk-fleet-management | 2021-11-25 15:24:34,345 [main] ERROR
com.zaxxer.hikari.pool.HikariPool - HikariPool-1 - Exception during pool initialization.
edge-server-sdk-fleet-management | org.postgresql.util.PSQLException: Connection to
localhost:5432 refused. Check that the hostname and port are correct and that the postmaster is
accepting TCP/IP connections.
edge-server-sdk-fleet-management |      at
org.postgresql.core.v3.ConnectionFactoryImpl.openConnectionImpl(ConnectionFactoryImpl.java:303)
edge-server-sdk-fleet-management |      at
org.postgresql.core.ConnectionFactory.openConnection(ConnectionFactory.java:51)
edge-server-sdk-fleet-management |      at
org.postgresql.jdbc.PgConnection.<init>(PgConnection.java:223)
edge-server-sdk-fleet-management |      at org.postgresql.Driver.makeConnection(Driver.java:465)
edge-server-sdk-fleet-management |      at org.postgresql.Driver.connect(Driver.java:264)
```

  If, after waiting for some time, the server is not up and running, and the server URL localhost:9090 is not showing the server page, uninstall and reinstall the playbook:

```
ansible-playbook AMR_server_containers/01_docker_sdk_env/docker_orchestration/ansible-playbooks/
02_edge_server/fleet_management/fleet_management_playbook_uninstall.yaml
ansible-playbook AMR_server_containers/01_docker_sdk_env/docker_orchestration/ansible-playbooks/
02_edge_server/fleet_management/fleet_management_playbook_install.yaml
```

  Result: The database is reset to the preconfigured database provided by Intel.

Fleet Management Server Dashboard over LAN Issues

If the Dashboard is not accessible from the client, the first step is to make sure that the client and server nodes are in the same subnet. This helper page can be used to find out: https://www.meridianoutpost.com/resources/etools/network/two-ips-on-same-network.php

If the client and server are in the same subnet, then it is possible that you are using proxies that prevent the connection. To check this on Linux, run the following command:

```
wget -q -T 3 -t 3 --no-proxy http://<IP>:9090/ && echo "COMMAND PASSED"
```

Where `<IP>` is the IP of your server.

If `COMMAND PASSED` is displayed, then you should configure your browser to NOT use proxy when accessing the IP/hostname of the server.

Playbook Install Errors

If you start the basic fleet management server right after a server reboot, you may encounter the error:

```
fatal: [localhost]: FAILED! => {"changed": false, "msg": "Logging into 10.237.22.88:30003 for
user admin
failed - 500 Server Error for http+docker://localhost/v1.41/auth: Internal Server Error
(\"Get \"https://10.237.22.88:30003/v2/\": dial tcp 10.237.22.88:30003: connect: connection
refused\")"}
```

1. Wait two minutes until the server is up and running.
2. Verify that all pods are running and no errors are reported:

```
kubectl get all -A
```

3. After all pods and services are up and running, restart the basic fleet management server:

```
ansible-playbook AMR_server_containers/01_docker_sdk_env/docker_orchestration/ansible-playbooks/
02_edge_server/fleet_management/fleet_management_playbook_install.yaml
```

Battery Status Not Available in Dashboard

To verify that the battery is correctly reported by the robot, check it on the client side:

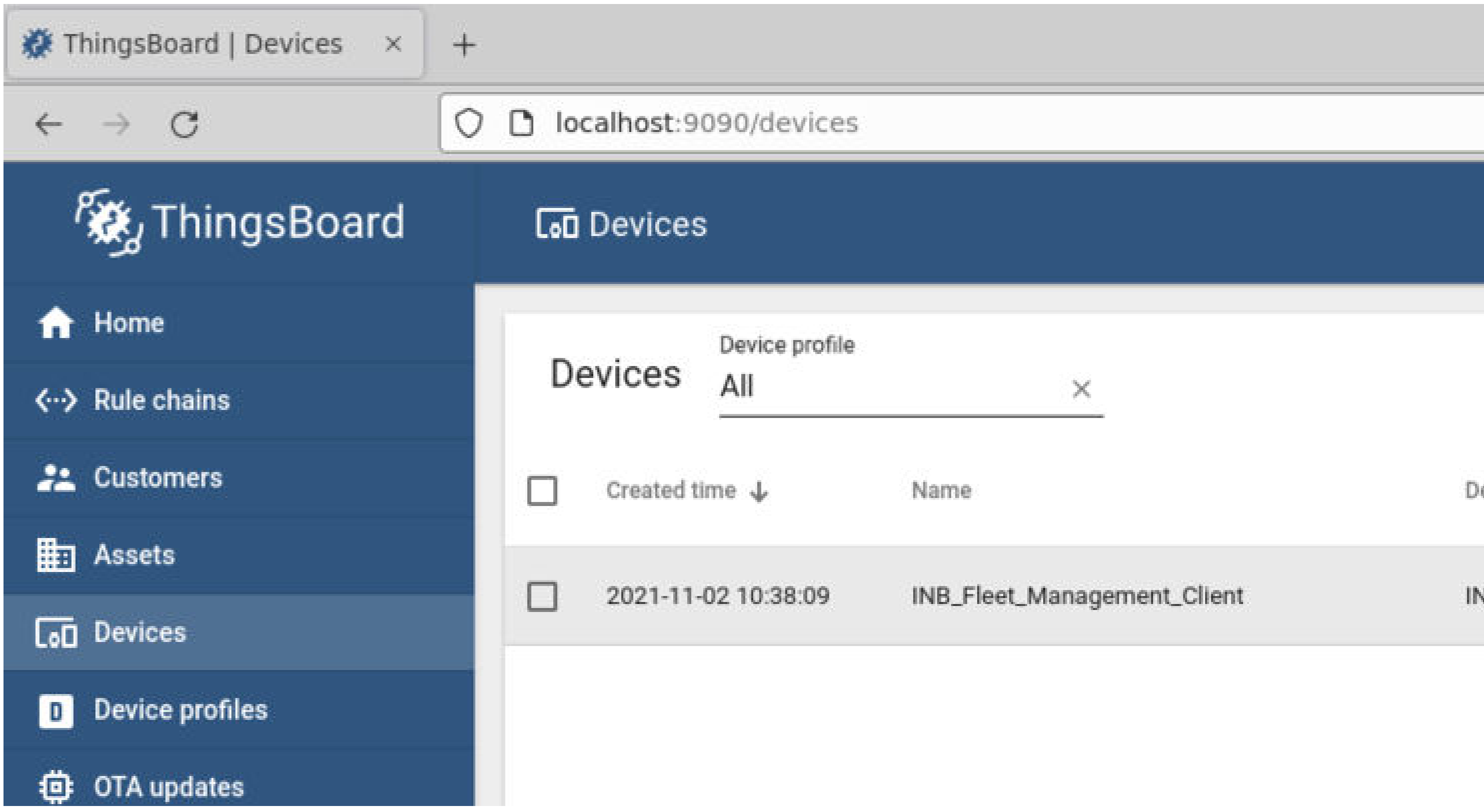
```
python
>>> import psutil
>>> battery = psutil.sensors_battery()
>>> print("Battery percentage : ", battery.percent)
Battery percentage : 43
```

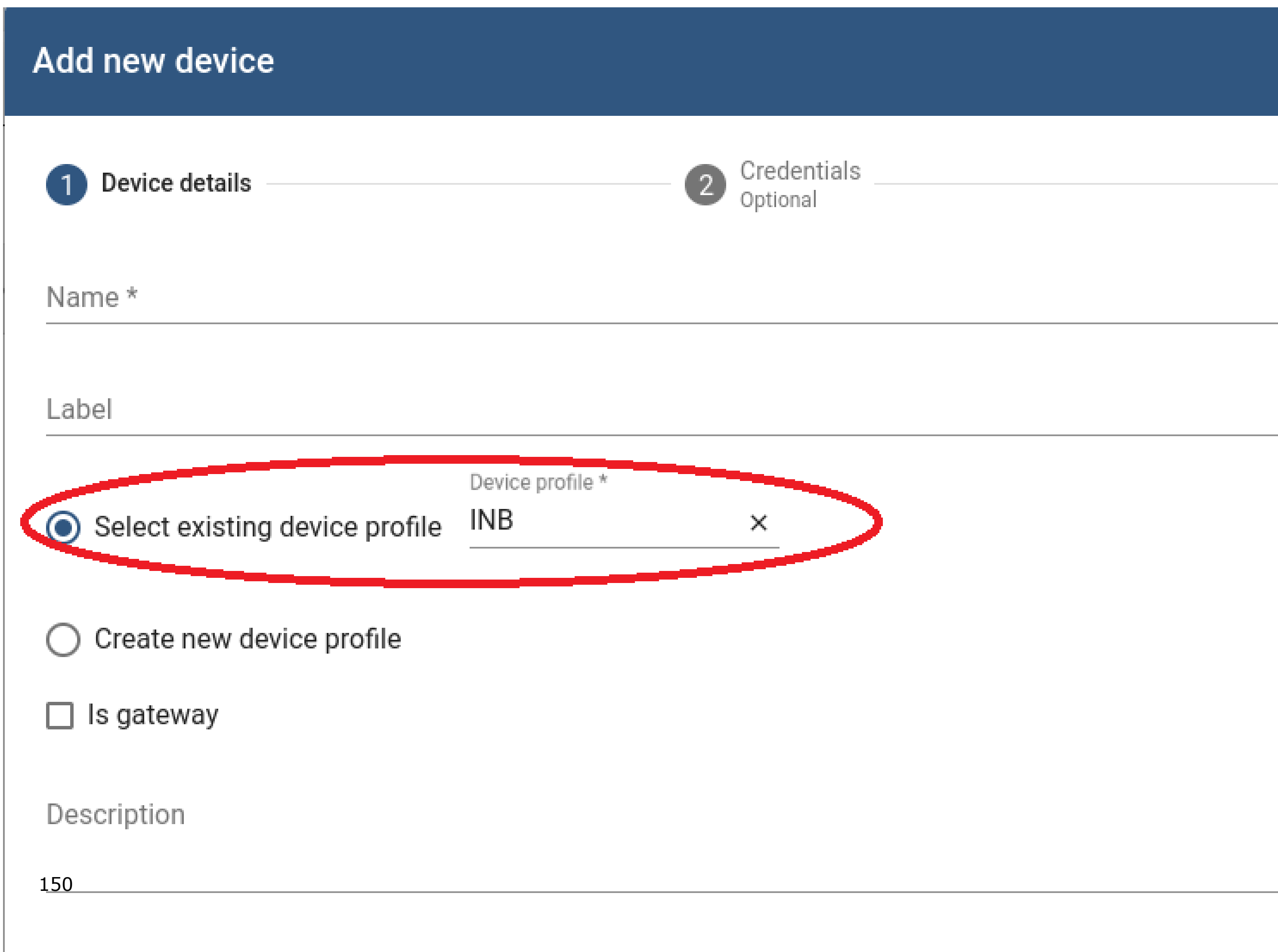
When the battery bridge is installed in robot, the 2 commands below are equivalent. So when you launch kobuki node, it publishes battery percentage in topic /sensors/battery_state. You can also do the same using the `ros2 topic pub` command.

```
# Publish battery status
ros2 topic pub /sensors/battery_state  sensor_msgs/msg/BatteryState "{percentage: 10}"
# or
launch kobuki image
source ros_entrypoint.sh
ros2 launch kobuki_node kobuki_node-composed-launch.py
```

Add New Clients to the Fleet Management Server

New devices can be created when more basic fleet management clients are going to be deployed. Remember to specify the Device Profile with which to associate the Device. It impacts the associated Rule Chain too.


ThingsBoard | Devices
localhost:9090/devices
ThingsBoard
Devices
Home
Rule chains
Customers
Assets
Devices
Device profiles
OTA updates
Devices
Device profile
All
Created time
Name
2021-11-02 10:38:09
INB_Fleet_Management_Client


Add new device
1 Device details
2 Credentials
Optional
Name *
Label
Device profile *
Select existing device profile
INB
Create new device profile
Is gateway
Description

For configuring new basic fleet management clients (1-to-1 mapping), the new tokens of the new Devices can be retrieved with **Copy access token**.

battery-bridge-kernel-module Install Failure

Follow the steps below:

```
cd components/amr_battery_bridge_kernel_module/src/
# uninstall battery-bridge-kernel-module
sudo ./module_install.sh -u
# check if below path exists
ls /sys/class/power_supply/BAT0
```

If the above path exists, then there is another kernel module occupying the place already and provided battery-bridge-kernel-module can not be installed.

In this case, the provided solution does work.

Pod Remains in "Terminating" State after Uninstall

After uninstall, if the pod does not stop but remains in a "Terminating" state, enter the following commands:

```
kubectl get pods -n ovms-tls
kubectl delete -n <pod_name_from_above_command> --grace-period=0 --force
ansible-playbook AMR_server_containers/01_docker_sdk_env/docker_orchestration/ansible-playbooks/
02_edge_server/openvino_model_server/ovms_playbook_uninstall.yaml
```

Intel® Edge Software Device Qualification (Intel® ESDQ) for EI for AMR

Overview

Intel® Edge Software Device Qualification (Intel® ESDQ) for EI for AMR provides customers with the capability to run an Intel provided test suite at the target system, with the goal of enabling partners to determine their platform's compatibility with the EI for AMR.

The target of this self certification suite is the EI for AMR compute systems. These platforms are the brain of the Robot Kit. They are responsible to get input from sensors, analyze them, and give instructions to the motors and wheels to move the EI for AMR.

How It Works

The EI for AMR Test Modules interacts with the Intel® ESDQ CLI through a common test module interface (TMI) layer which is part of the Intel® ESDQ binary. Intel® ESDQ generates a complete test report in HTML format, along with detailed logs packaged as one zip file, which you can manually choose to email to: Edge.Software.Partners@intel.com

> **NOTE** Each test and its pass/fail criteria is described below. To jump to the installation process, go to Download and Install Intel® ESDQ for EI for AMR.

Intel® ESDQ for EI for AMR contains the following test modules.

- Docker* Container

  This module verifies that the EI for AMR comes as a Docker* container and it can run on the target platform.

For more information, go to the Docker* website.

The test is considered Pass if:

- The Docker* container can be opened.

- Intel® RealSense™ Camera

  This module verifies the capabilities of the Intel® RealSense™ technology on the target platform.

  For more information, go to the Intel® RealSense™ website.

  The tests within this module verify that the following features are installed properly on the target platform and that EI for AMR and the Intel® RealSense™ camera are functioning properly:

  - The camera is detected and is working.
  - Intel® RealSense™ SDK.

  The tests are considered Pass if:

  - The Intel® RealSense™ SDK 2.0 libraries are present in Docker* container.
  - A simple C++ file can be compiled using g++ and -lrealsense2 flag.
  - Intel® RealSense™ Topics are listed and published.
  - The number of FPS (Frames Per Second) are as expected.
- Intel® VTune™ Profiler

  This module runs the Intel® VTune™ Profiler on the target system.

  For more information, go to the Intel® VTune™ Profiler website.

  The test is considered Pass if:

  - VTune™ Profiler runs without errors.
  - VTune™ Profiler collects Platform information.
- rviz2 and Fast Mapping

  This module runs the Fast Mapping application (the version of octomap optomized for Intel) on the target system and uses rviz2 to verify that it works as expected.

  For more information, go to the rviz wiki.

  The test is considered Pass if:

  - Fast Mapping is able to create a map out of a pre-recorded ros2 bag.
- Turtlesim

  This module runs the Turtlesim ROS2 application on the target system and checks if it works as expected.

  For more information, go to the Turtlesim wiki.

  The test is considered Pass if:

  - Turtlesim opens and runs without error.
- Intel® oneAPI Base Toolkit

  This module verifies some basic capabilities of Intel® oneAPI Base Toolkit on the target platform.

  For more information, go to the Intel® OneAPI Base Toolkit website.

  The tests within this module verify that the following features are functioning properly on the target platform:

  - DPC++ compiler
  - CUDA to DPC++ converter

  This test is considered Pass if:

  - A simple C++ file can be compiled using the DPC++ compiled and it runs as expected.
  - CUDA can be installed.

  - A CUDA specific file can be converted to DPC++ and it runs as expected.
- OpenVINO™ Toolkit

  This module verifies two core features of the OpenVINO™ Toolkit:

  - OpenVINO™ model optimizer
  - Object detection using TensorFlow*

  The test is considered Pass if:

  - The OpenVINO™ model optimizer is capable to transform a TensorFlow model to an Intermediate Representation (IR) of the network, which can be inferred with the Inference Engine.
- Object Detection on CPU

  This module verifies object detection using OpenVINO™ on CPU.

  The test is considered Pass if:

  - The object is detected.

  If the test is failed, you can check the expected picture and the actual picture obtained by the test.
- Object Detection on VPU

  This module verifies object detection using OpenVINO™ on VPU.

  The test is considered Pass if:

  - The object is detected.

  If the test is failed, you can check the expected picture and the actual picture obtained by the test.
- Object Detection on Intel® Movidius™ Myriad™ X VPU

  This module verifies object detection using OpenVINO™ on Intel® Movidius™ Myriad™ X VPU.

  The test is considered Pass if:

  - The object is detected.

  If the test is failed, you can check the expected picture and the actual picture obtained by the test.
- GStreamer* Video

  This module verifies if a GStreamer* Video Pipeline using GStreamer* Plugins runs on the target system.

  The test is considered Pass if:

  - The Video Pipeline was opened on the host without errors.
- GStreamer* Audio

  This module verifies if a GStreamer* Audio Pipeline using GStreamer* Plugins runs on the target system.

  The test is considered Pass if:

  - The Audio Pipeline was opened on the host without errors.
- GStreamer* Autovideosink Plugin - Display

  This module verifies if a stream from a camera compatible with libv4l2 can be opened and displayed using GStreamer*.

  The test is considered Pass if:

  - No Error messages are displayed while running the gst-launch command.

  This test may FAIL, or it may be skipped if the target system does not have a Web Camera connected.
- GStreamer* Intel® RealSense™ Video Plugin

  This module verifies if a GStreamer* Video Pipeline using the Intel® RealSense™ Plugin runs on the target system.

  The test is considered Pass if:

- No Error messages are displayed while running the gst-launch command.

  This test may FAIL, or it may be skipped if the target system does not have a Intel® RealSense™ Camera connected.
- ADBSCAN

  This module verifies if the ADBSCAN algorithm works on the target system.

  The test is considered Pass if:

  - The ADBSCAN algorithm works on the target system.
- Collaborative SLAM

  This module verifies if the collaborative SLAM algorithm works on the target system.

  The test is considered Pass if:

  - The collaborative SLAM algorithm works on the target system.

Get Started

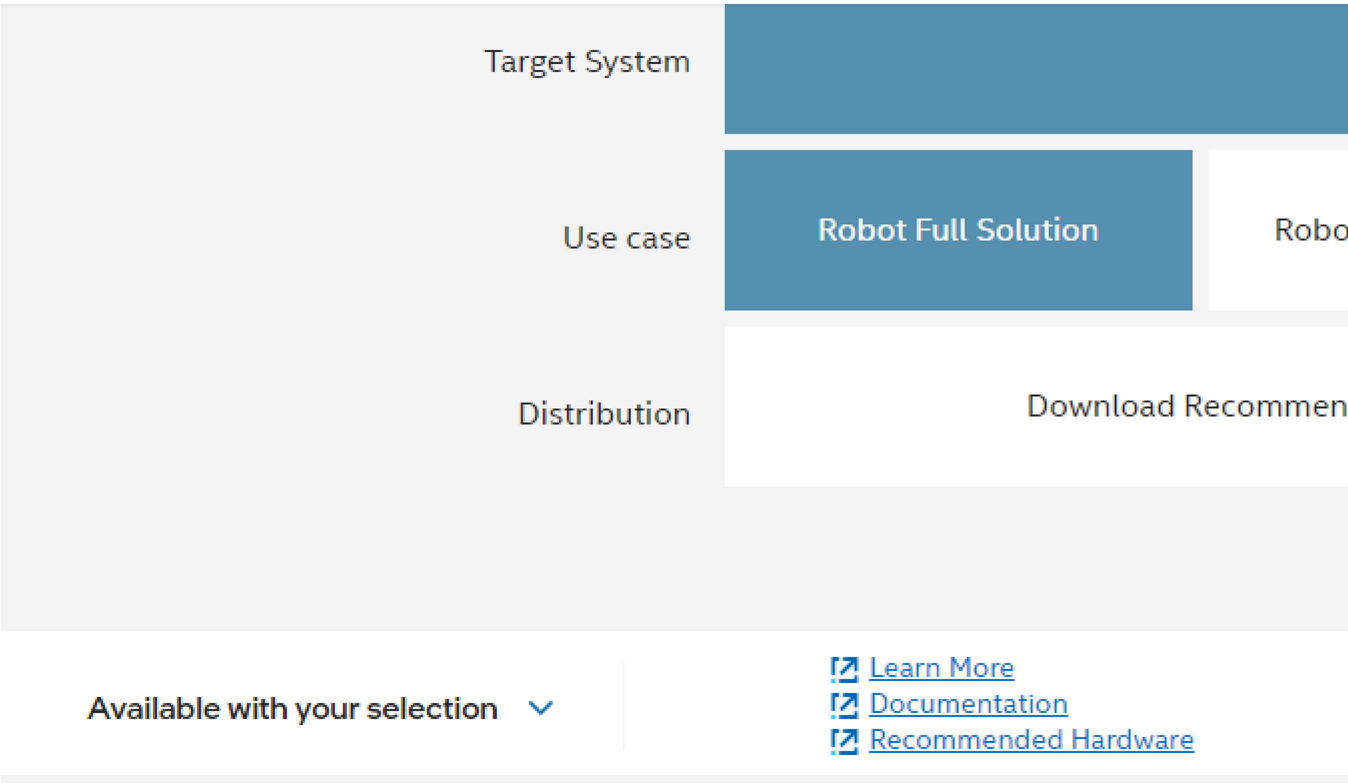
This step-by-step guide takes you through installing the Intel® ESDQ CLI tool, which is installed as part of the EI for AMR. Refer to the How It Works section before you get started with the installation. To use these instructions, you must download the Edge Insights for Autonomous Mobile Robots package. You can download the default packages or you can customize the package download depending upon your needs.

Download and Install Intel® ESDQ for EI for AMR

Intel® ESDQ is optionally bundled with EI for AMR solutions.

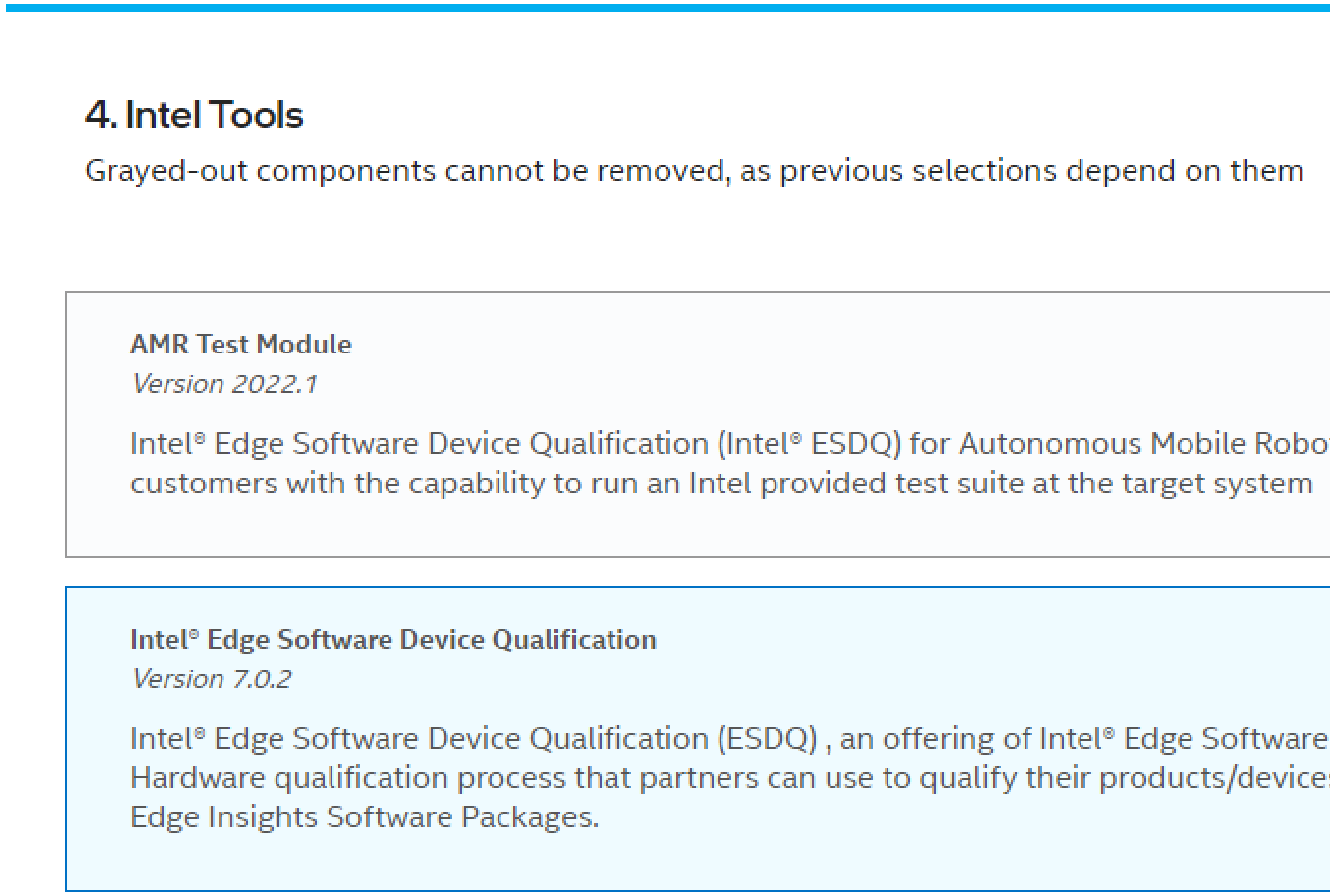
1. Download a configuration that includes Intel® ESDQ.

   a. Go to the Product Download page.
   b. Select **Robot Complete Kit**, **Server Complete Kit**, or **Robot and Sever Complete Kit**.
   c. Select **Customize Download**.



d. Click **Next**, until you get to step 4.

e. On the **Reference Implementations** page, make sure that **Intel® Edge Software Device Qualification** is checked.



f. Click **Next** until you get to the **Download** page, and click on **Download**.

2. Follow the steps in the Get Started Guide for Robots to extract and install EI for AMR.

Run the Application

1. Change the directory:

```
cd $HOME/edge_software_device_qualification/Edge_Software_Device_Qualification_For_AMR_*/esdq
```

2. Unzip the ROS 2 bags used in the tests:

```
unzip ../AMR_containers/01_docker_sdk_env/docker_compose/06_bags.zip -d ../AMR_containers/
01_docker_sdk_env/docker_compose/
```

3. Run the Intel® ESDQ test, and generate the report:

```
./esdq run -r
```

Expected output (These results are for illustration purposes only.)

Summary

	Property	Value
0	ESDQ Version	6.0
1	Execution Timestamp UTC	2021-12-14 19:52:26
2	Duration	0:31:21
3	Number of test category	2
4	Test Categories	'SysInfo' 'AMR_Test_Module'

SystemInfo

	System Category	Property	Value
0	SOFTWARE INFO	OS version	Ubuntu 20.04.3 LTS
1	SOFTWARE INFO	Kernel	5.4.0-89-generic
2	HARDWARE INFO	Device Manufacturer	Intel Corporation
3	HARDWARE INFO	Hardware Architecture	x86_64
4	HARDWARE INFO	Processor	Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
5	HARDWARE INFO	GPU	UHD Graphics 630 (Mobile)
6	HARDWARE INFO	Memory size	31Gi
7	HDD Configurations	Model	ATA CT1000MX500SSD4 (scsi)
8	HDD Configurations	Disk /dev/sda	1000GB
9	HDD Configurations	Sector size (logical/physical)	512B/4096B
10	HDD Configurations	Partition Table	gpt
11	HDD Configurations	Disk Flags Header	Number Start End Size File system Name Flags
12	HDD Configurations	Disk Flags	1 1049kB 538MB 537MB fat32 EFI System Partition boot esp
13	HDD Configurations	Disk Flags	2 538MB 1000GB 1000GB lvm
14	HDD Configurations	Model	Linux device-mapper (linear) (dm)
15	HDD Configurations	Disk /dev/mapper/vgglaic3n144-swap_1	1023MB
16	HDD Configurations	Sector size (logical/physical)	512B/4096B
17	HDD Configurations	Partition Table	loop
18	HDD Configurations	Disk Flags Header	Number Start End Size File system Flags
19	HDD Configurations	Disk Flags	1 0.00B 1023MB 1023MB linux-swap(v1)
20	HDD Configurations	Model	Linux device-mapper (linear) (dm)
21	HDD Configurations	Disk /dev/mapper/vgglaic3n144-root	999GB
22	HDD Configurations	Sector size (logical/physical)	512B/4096B
23	HDD Configurations	Partition Table	loop
24	HDD Configurations	Disk Flags Header	Number Start End Size File system Flags
25	HDD Configurations	Disk Flags	1 0.00B 999GB 999GB ext4
26	HARDWARE ACCELERATOR	VPU	0
27	HARDWARE ACCELERATOR	FPGA	0

ModulesInfo

	Installed Module	Status
0	Docker_Community_Edition_CE	SUCCESS
1	Docker_Compose	SUCCESS
2	esdq	SUCCESS

Test Suites

AMR_Test_Module

	ModuleName	Test CaseName	Status
0	AMR1	System requirements	Pass
1	AMR2	OneAPI	Pass
2	AMR3	OpenVINO	Pass

NOTE The OpenVINO™ Object Detection Myriad Test Failure above is shown for demonstration purposes only. The test is expected to pass.

Send Results to Intel

Once all the automated and manual tests are executed successfully, you can submit your test results and get your devices listed on the Intel® Edge Software Recommended Hardware site.

Send the zip file that is created after running Intel® ESDQ tests to: Edge.Software.Partners@intel.com.

For example, after one of our local runs the following file was generated:
`esdqReport_2022-03-09_13:22:28.zip`

Troubleshooting

For issues, go to: Troubleshooting for Robot Tutorials.

Support Forum

If you're unable to resolve your issues, contact the Support Forum.

Security

This section highlights the security features offered by the Edge Insights for Autonomous Mobile Robots (EI for AMR) platform and provides an overview of the security features. For further reading, refer to the specific documents listed below.

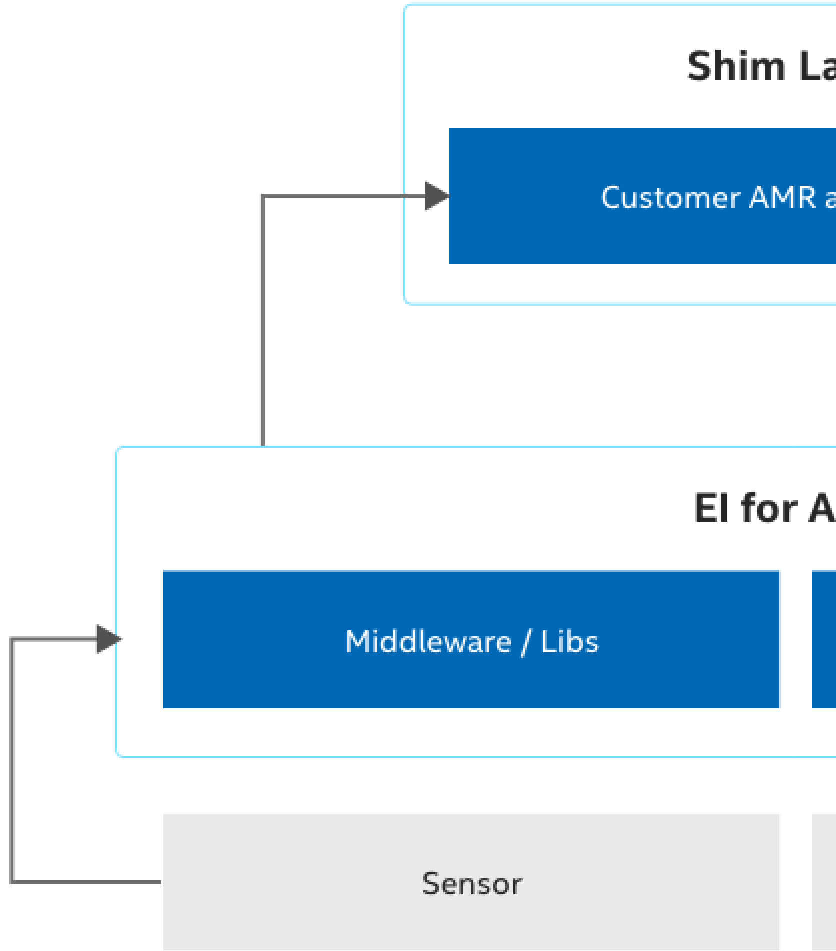
Shim Layer - Protect your application data

The EI for AMR includes open-source components, which may be affected by vulnerabilities. A shim layer can help to protect your program data against an attack initiated via these vulnerabilities.

The main task of the shim layer is to reduce the attack surface by verifying the data (such as size, value range, memory range, etc.) transferred via a function call to or from a library or an executable and protect the customer code and data via this mechanism.

Due to architecture constraints, only the developer of the application code can implement the matching shim layer correctly.

The following picture shows a potential implementation of a shim layer around the customer application like a shell.


Shim La
Customer AMR a
EI for A
Middleware / Libs
Sensor

Keep in mind, complex checks and more layers might have an impact on the overall system performance.

In general, it is highly recommended to check regularly for updates and vulnerabilities on the component web sites.

Edge Insights for Autonomous Mobile Robots Platform

The main EI for AMR platform is based on the 11th generation Intel® Core™ processor with accelerators primarily used for AI inference and vision processing. The platform inherits many security elements from the processor.

Security Use Cases and Features

The EI for AMR platform offers various security features that customers can leverage in the context of Autonomous Robotics Applications. They are listed as follows:

Secure Boot

Ensure the system boots from a trusted source and is not manipulated by an attacker. To establish a secure boot, a chain of trust is set up; the root-of trust is unmodifiable by nature. Typically, the root-of trust is a key burned in fuses in the device or ROM based program code.

Intel devices support secure boot with Intel® Trusted Execution Technology (Intel® TXT) and offers via the Intel® CSME, a software implementation of the Trusted Platform Module (TPM).

More information about the described use cases and features can be found in the following documents:

Document Title	Intel Document ID	Document Link
Intel® Converged Boot Guard and Intel® Trusted Execution Technology (Intel® TXT)	575623	https://cdrdv2.intel.com/v1/dl/getContent/575623
Tiger Lake platform - Firmware Architecture Specification	608531	https://cdrdv2.intel.com/v1/dl/getContent/608531
Intel® Trusted Execution Technology (Intel® TXT) DMA Protection Ranges	633933	https://cdrdv2.intel.com/v1/dl/getContent/633933
Intel® Trusted Execution Technology (Intel® TXT) Enabling Guide	–	https://www.intel.com/content/www/us/en/developer/articles/guide/intel-trusted-execution-technology-intel-txt-enabling-guide.html
Trusted Platform Module Specification	–	https://trustedcomputinggroup.org/

Authentication

Authentication helps to develop a secure system. A run-time authentication system is the next step following secure boot. Any program code can be authenticated before it is executed by the system. This powerful tool enables AMR suppliers to guarantee a level of security, and safety during run-time. Executing code from an unknown source or malware wouldn't be possible.

The Intel® Dynamic Application Loader (Intel® DAL) is a feature of Intel® platforms that allows you to run small portions of Java* code on Intel® Converged Security and Management Engine (Intel® CSME) firmware. Intel has developed DAL Host Interface Daemon (also known as JHI), which contains the APIs that enable a Linux* operating system to communicate with Intel DAL. The daemon is available both in a standalone software package and as part of the Linux* Yocto 64-bit distribution.

More information about the described use cases and features can be found in the following documents:

Document Title	Intel Document ID	Document Link
Trusty TEE Software Architecture Specification	607736	https://cdrdv2.intel.com/v1/dl/getContent/607736
Intel® Dynamic Application Loader (Intel® DAL) Developer Guide	–	https://www.intel.com/content/www/us/en/develop/documentation/dal-developer-guide/top.html

Virtualization

Virtualization is another important element to increase the level of security and safety. It helps to establish freedom from interference (FFI), as it's requested for safety use cases, and workload consolidation. Intel devices have supported this use case with Intel® Virtualization Technology (Intel® VT) for decades.

More information about the described use cases and features can be found in the following documents:

Document Title	Intel Document ID	Document Link
Intel® 64 and IA-32 Architectures Software Developer Manuals	–	https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html

Encryption

Encryption is required for many security use cases. The EI for AMR platform supports the common encryption algorithms like AES or RSA in hardware. This increases the encryption/decryption performance and the security level. Typical use cases are the encryption of communication messages, a file system, or single files for IP protection or the creation of a secure storage for security relevant data like crypto keys or passwords. Another use case is memory encryption; the EI for AMR platform supports this with the Total Memory Encryption (TME) feature.

More information about the described use cases and features can be found in the following documents:

Document Title	Intel Document ID	Document Link
Tiger Lake platform Intel® Total Memory Encryption (Intel® TME)	620815	https://cdrdv2.intel.com/v1/dl/getContent/620815
Whitley Platform Memory Encryption Technologies -TME/MK-TME deep Dive	611211	https://cdrdv2.intel.com/v1/dl/getContent/611211
Intel® 64 and IA-32 Architectures Software Developer Manuals	–	https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html
Filesystem-level encryption (Linux*)	–	https://www.kernel.org/doc/html/v4.16/filesystems/fscrypt.html
Intel® Advanced Encryption Standard Instructions (AES-NI)	–	https://www.intel.com/content/dam/develop/external/us/en/documents/introduction-to-intel-secure-key-instructions.pdf

Firmware Update

To improve the security and safety status over the lifetime of a device, the internal firmware (e.g. BIOS) must be updatable. In this case the update packages are signed by the supplier (e.g. Intel, OEM etc.).

More information about the described use cases and features can be found in the following document:

Document Title	Intel Document ID	Document Link
Tiger Lake platform - Firmware Architecture Specification	608531	https://cdrdv2.intel.com/v1/dl/getContent/608531

Secure Debug

Debugging is an important feature during product development. During in-field usage, debugging might also be needed to analyze field returns. To prevent anyone from accessing internal resources via the debugger, a secure debugging system is developed. In this case an engineer who wants to use the debugger has to authenticate via a valid token which has to be offered to the system (e.g. storing it in flash). Tokens must be signed by a key which was stored during manufacturing flow into the device fuses.

More information about the described use cases and features can be found in the following documents:

Document Title	Intel Document ID	Document Link
Tiger Lake platform - Firmware Architecture Specification	608531	https://cdrdv2.intel.com/v1/dl/getContent/608531
2019 Spring Client Customer Debug Methodologies and Tools	612942	https://cdrdv2.intel.com/v1/dl/getContent/612942
Tiger Lake platform enDebug User Guide	630604	https://cdrdv2.intel.com/v1/dl/getContent/630604
Anderson Lake Secure Debug User Guide	614222	https://cdrdv2.intel.com/v1/dl/getContent/614222

Docker* Installation and Usage

Docker* is not a primary use case of EI for AMR systems. Docker uses virtualization on the OS-level to deliver software in packages called *containers*. The EI for AMR package is delivered in this form. It is up to you to decide whether or not to re-use this approach in the final product.

The host system owner can improve the security level of the Docker installation and Docker during run-time. For this, it is useful to check if your system follows Docker best practices.

Additionally, it would be good to check your Docker installation with the CIS Docker benchmark. This benchmark checks several aspects of the installation and run-time configuration and gives you a good indication of improvements.

Real-Time Support

Intel real-time technology supports new solutions that require a high degree of coordination, both within and across network devices. Intel® Time Coordinated Computing (Intel® TCC)-enabled processors deliver optimal compute and time performance for real-time applications. Using integrated or discrete Ethernet controllers featuring IEEE 802.1 Time Sensitive Networking (TSN), these processors can power complex real-time systems.

For more information, refer to:

- Intel Real-Time Computing IoT Technology Resources

- Intel® Time Coordinated Computing Tools
- Intel IoT Real-Time Technical Library

Terminology

Term	Description
ADBSCAN	Adaptive Density-Based Spatial Clustering of Applications with Noise
AOTA	Application Over the Air
ARIAC	Agile Robotics for Industrial Automation Competition
CNDA	Corporate Non-Disclosure Agreement
CPU	Central Processing Unit
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DI	Device Initialization Protocol
DL	Deep Learning
DMS	Device Management Service
DPC++	Data Parallel C++
DRM	Deterministic Road Map
EI for AMR	Edge Insights for Autonomous Mobile Robots
DPC++	Data Parallel C++
EOF	end-of-file
FDO	FIDO Device Onboard
FIDO	Fast IDentity Online
FLANN	Fast Library for Approximate Nearest Neighbors
FM	Fast Mapping
GEAR	Gazebo Environment for Agile Robotics
GPU	Graphics Processor Unit
GSLAM	General Simultaneous Localization and Mapping
GUI	Graphical User Interface
IDE	Integrated Development Environment
IE	Inference Engine
IMU	Inertial Measurement Unit
IPU	Image Processing Unit
ITS	Intelligent sampling and Two-way Search

Term	Description
KVM	Kernel-based Virtual Machine
LIDAR	Light Detection and Ranging
MQTT	Message Queuing Telemetry Transport
NFS	Network File System
NN	Neural Network
OSRF	Open Source Robotics Foundation
OTA	Over-The-Air
PCL	Point Cloud Library
PRM	Probabilistic Road Map
RDC	Resource and Documentation Center
RGBD	Red, Green, Blue plus Depth
ROS	Robot Operating System
RPLIDAR	360-degree 2D LIDAR solution developed by SLAMTEC
RPM	Red Hat* Package Manager
RTAB-Map	Real-Time Appearance-Based Mapping
RV	Rendezvous
SDK	Software Development Kit
SDO	Intel® Secure Device Onboard (Intel® SDO)
SLAM	Simultaneous Localization And Mapping
SOTA	Software Over the Air
SSD	Single-Shot multibox Detection
SSL	Secure Sockets Layer
TLS	Transport Layer Security
TMI	Test Module Interface
UEFI	Unified Extensible Firmware Interface
VNC	Virtual Network Computing
vSLAM	Visual Simultaneous Localization and Mapping

Notices and Disclaimers